Bruk av RDF til kobling av RFID baserte metadata til kvalitetssikring hos Agder Renovasjon

av

Kjartan Gausland
Johannes Sørheim

Masteroppgave i
informasjons- og kommunikasjonsteknologi

Høgskolen i Agder
Fakultet for teknologi

Grimstad
mai 2005

Sammendrag

Resource Description Framework (RDF) er et rammeverk som blir standardisert av W3C. Rammeverket brukes til å beskrive og koble sammen ressurser ved hjelp av subjekt, egenskap og objekt. I denne oppgaven har vi benyttet rammeverket i kombinasjon med radiofrekvens identifisering (RFID) for å se hvilke fordeler og ulemper rammeverket har i forhold til relasjonsdatabaser. Som praktisk case har vi sett på hvordan RDF og RFID kan brukes til kvalitetssikring hos Agder Renovasjon.

For å kunne gi en god vurdering har vi gjort en studie av hva RDF er og hvordan rammeverket fungerer. Deretter har vi sett på ulike RDF verktøy som kan benyttes til utvikling av en demonstrator. Det finnes mange RDF løsninger, men de fleste av disse har blitt utviklet i forbindelse med forskningsprosjekter og er kun tilgjengelige i relativt ufullstendige versjoner. Etter vurderinger av en del RDF verktøy endte vi opp med to alternativer: Sesame og Jena. Disse to er ganske likeverdige, men vi valgte Jena til utviklingen av en demonstrator siden vi hadde brukt dette mest.

Hensikten med å benytte RDF er at dette er et enkelt rammeverk for å beskrive ressurser. Ressursene kan ha varierende antall egenskaper, og får globalt unike identifikatorer. Med RDF blir det enkelt å integrere flere modeller. Vår erfaring med rammeverket er at det fungerer greit, men får problemer med tidsbruk ved store modeller. Dette ser ut til å gjelde for de fleste RDF rammeverkene.

I tillegg til RDF har vi sett på ulike alternativer til RFID utstyr. Vi har sett på dette med tanke på å finne utstyr til utviklingen av en demonstrator. Til demonstratoren valgte vi å benytte bærbart utstyr, for å gjøre det enklest mulig å flytte utstyret mellom HiA og renovasjonsbilen. RFID leseren har fungert greit til dette formålet, men i et endelig system ville vi foretrukket stasjonært RFID utstyr montert på bilene. Det ble ikke brukt spesielle RFID brikker, men standard brikker i kredittkortstørrelse. Disse ble festet under lokket på papirdunkene på testruten.

Ved å merke avfallsdunker ved hjelp av RFID brikker, og benytte et system som registrerer informasjon om blant annet når dunkene blir tømt, vil vi få dokumentert det arbeidet som blir utført. Dette vil også gi muligheter for å varsle sjåføren dersom en dunk blir glemt, slik at feil kan rettes opp så tidlig som mulig. Kvalitetssikringssystemet skal kunne vise informasjon som kan brukes til optimalisering av rutene, og brukes som dokumentasjon i forbindelse med klager fra kunder.

Sett i en større sammenheng ser vi at RDF egner seg best der det kreves et felles grensesnitt mellom ulike kilder. RDF kan være nyttig ved integrering mot eksisterende systemer, eller til sammenslåing av informasjon fra flere selskap for oppretting av nasjonale statistikker.

Forord

Denne masteroppgaven ble skrevet for SMARTsolutions AS i Grimstad. Oppgaven er avslutningen på en femårig masterutdannelse i informasjons- og kommunikasjonsteknologi (Master IKT) ved Høgskolen i Agder, avdeling Grimstad. Arbeidet er utført i Grimstad i perioden januar til mai 2005 og teller 30 studiepoeng.

Vi vil først og fremst få takke SMARTsolutions og Agder Renovasjon for godt samarbeid under prosjektperioden. Vi vil også takke David Norheim i @semantics for nyttig informasjon om RDF.

Til slutt vil vi takke vår veileder på HiA, Høgskolelektor Magne Arild Haglund, for god veiledning og gode innspill.

Grimstad, 30. mai 2005

Kjartan Gausland

Johannes Sørheim

Innholdsfortegnelse

Figur- og tabellister

Forkortelser

RDF – Resource Description Framework
RFID – Radio Frequency Identification
W3C – World Wide Web Consortium
N3 – Notation 3
Nt – N-Triples
XML – Extensible Markup Language
URI – Uniform Resource Identifier
VPN – Virtual Private Network
API – Application Programming Interface
BLOB – Binary Large Object

1 Innledning

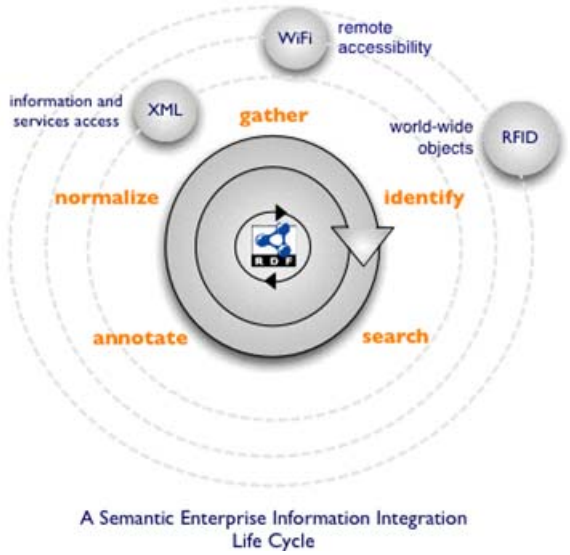
1.1 Bakgrunn

Dette studiet skal se på hvordan Resource Description Framework (RDF) og Enterprise Information Integration kan brukes til hensiktsmessig håndtering og integrering av metadata. Vi skal se på dette i et konkret tilfelle der metadata blir benyttet til kvalitetssikring i renovasjonsbransjen. Her skal vi benytte RFID til identifisering av søppeldunker, og knytte metadata til disse.

SMARTsolutions AS er et ide og produktutviklingsselskap innen IKT. Bakgrunnen for dette studiet er et ønske fra SMARTsolutions om å bygge opp kunnskap innen maskin til maskin kommunikasjon, RDF og Enterprise Information Integration. Dette vil gi dem et større vurderingsgrunnlag ved valg av teknologi til håndtering av metadata. I fellesskap med oppdragsgiver har vi kommet frem til et konkret anvendelsesområde vi skal se nærmere på.

Vi hadde et møte med SMARTsolutions og daglig leder i Agder Renovasjon. Her kom det frem at Agder Renovasjon ønsker å øke kvaliteten på tjenestene sine. I dag får de en del klager fra kunder som mener at avfallet ikke er hentet. I en del tilfeller stemmer dette, men i andre tilfeller klager kunden selv om avfallet er hentet. I begge tilfeller vil det være en fordel for Agder Renovasjon å ha et system der hver enkelt tømming blir registrert med tidspunkt, hvem som tømte og lignende.

I Kristiansand kommune har de innført et system som registrerer hver tømming, og fakturerer ut fra antall tømminger. Dette systemet bruker RFID til å identifisere avfallsdunkene. Dette studiet har et annet fokus, ved at vi ser på muligheten til å benytte RFID til kvalitetssikring av tømmingen. Hovedfokus her er ikke RFID teknologien, men hvordan informasjonen knyttet til avfallsdunkene kan struktureres, lagres og integreres på en hensiktsmessig måte ved hjelp av RDF. RFID blir i dette tilfellet et verktøy som benyttes for å identifisere dunkene.



Figur 1 – Enterprise Information Integration

RDF er en W3C standard som brukes for å beskrive ressurser. RDF baserer seg på XML format, og har mange standard tagger for å beskrive ulike ressurser. RDF dataene kan lagres på flere ulike måter, som filer, i databaser eller datavarehus, og det finnes flere alternativer innenfor disse. Utvekslingen av RDF data kan skje via blant annet WebServices, http eller filer. Denne informasjonen kan samles sammen ved hjelp av Enterprise Information Integration.

Enterprise Information Integration (EII) er en tilnærming for å samle data fra ulike kilder. EII minimerer omfanget av endringer i kilde systemene og maksimerer forretningseffektiviteten. [1]

Ved å benytte en føderert datamodell som spenner over ressurser som kan unikt identifiseres, oppnår man noe man ikke kan oppnå med teknologiene alene: Et sammenkoblet oversiktsbilde over alle ressursene uavhengig av lagring, teknologi og egenskap. Dette er EII. [2]

1.2 Oppgavebeskrivelse

Agder Renovasjon ønsker gjennom oppgaven å forbedre kvalitetssikringen på tjenestene sine. SMARTsolutions ønsker å utvikle løsninger for dette.

Studentene skal se på hvordan RDF kan brukes for å koble sammen metadata slik at disse kan brukes til kvalitetssikring i en case hos Agder Renovasjon. Kvalitetssikring kan blant annet være dokumentering av data vedrørende tømming av avfallsdunker.

Studentene skal finne frem til noen RDF verktøy og sammenlikne disse for å finne et som er godt egnet til utvikling av en demonstrator.

Det er ønskelig at det også utvikles en demonstrator som ved hjelp av RFID demonstrerer noen av mulighetene til et kvalitetssikringssystem for renovasjonsbransjen.

Etter implementeringen av demonstratoren skal studentene se på muligheter, fordeler og ulemper ved bruken av en føderert datamodell som RDF i forhold til tradisjonelle databasesystemer.

1.3 Teoretisk rammeverk

RDF: Resource Description Framework er en W3C standard. RDF beskriver egenskapene til ressurser på en klar og utvetydig måte.

RFID: Radio Frequency Identification er en teknologi for å identifisere objekter ved hjelp av radiobølger. Det eksisterer forskjellige typer RFID brikker, både aktive og passive. Mange brikker har mulighet til å lagre begrensede mengder informasjon. Alle brikkene har en egen ID som gjør at disse kan identifiseres.

EII: Enterprise Information Integration er integrering av data på tvers av lagring, teknologi, egenskap og bruk.

Metadata: Metadata er data om data, eller data som beskriver en ressurs. I dette studiet vil metadata være dataene som er knyttet til en avfallsdunk ved hjelp av en RFID brikke. Dette kan for eksempel være: tidspunkt, kunde, adresse, rute, sjåfør eller liknende.

1.4 Litteraturstudier

Når det gjelder lagring av metadata i RDF format, beskriver [3] ulike strategier for hvordan dette kan gjøres med RDBMS. Karakteristikken av RDF er ganske tilsvarende XML, og de ser derfor på metoder for å lagre XML dokumenter i en relasjonsdatabase.

I [4] ser forfatterne på hvordan en kan transformere XML til RDF. De tar også for seg lagringsstruktur for RDF i releasjonsdatabaser. Ved å benytte RDF som har standard og repetert struktur på formatet: ressurs, egenskap og verdi, foreslår forfatterne å lagre dataene i relasjonsdatabaser ved hjelp av tabellene: ressurs, egenskap, verdi og klasse.

Artikkelen [5] ser på de samme tingene, men med en litt annen vinkling. Her fokuseres det på lagring, RDF spørring og skjema informasjon. Ulike implementeringer av lagring av metadata blir presentert, disse består av ulike typer databaser, ”RDF store” og RDF filer.

Når RDF blir brukt til representasjon av metadata, vil informasjonen lettere kunne utveksles mellom ulike datasystemer. Effekten av dette er ifølge [2] blant annet reduserte integreringskostnader og økt produktivitet.

Nyttig informasjon og referanser kan finnes i [6]. Dokumentet gir mange nyttige og bra referanser til standarder, vokabular, verktøy, programvare og demoer som er nyttig for RDF.

1.5 Omfang og Begrensninger

1.5.1 Omfang

Vi skal gjennomføre en grundig studie på bruk av RDF, der vi skal se på hensiktsmessig håndtering og integrering av metadata. Hva er fordeler og ulemper i forhold til alternative teknologier? I tillegg skal vi beskrive noen alternative teknologier som kan brukes til lagringen av metadata, for eksempel relasjons databaser eller datavarehus. Vi skal belyse i hvilken grad RFID i kombinasjon med RDF er relevant for Agder Renovasjon.

Vi skal også implementere en demonstrator som benytter RDF til håndtering av metadata. Denne demonstratoren skal kunne lese RFID tagger og legge informasjonen inn i systemet. Informasjon om skanningen inneholder blant annet tidspunkt, lokasjon, sjåfør osv. Demonstratoren skal også kunne gi tilbakemelding til sjåfør dersom en dunk glemmes. Informasjonen skal midlertidig ligge lokalt (i skanner eller søppelbil), og etter endt rute synkroniseres ved avfallsanlegget.

Demonstratoren skal brukes til å kjøre en praktisk testperiode der avfallsdunker på et byggefelt eller en gate merkes med RFID brikker, og sjåføren utstyres med utstyr til å skanne disse. Etter testperioden skal vi evaluere resultatet.

1.5.2 Begrensninger

Vi skal i dette studiet benytte RFID som et verktøy, og vil derfor ikke gå noe særlig inn på studier av RFID. Vi skal sette oss inn i det som trengs for å gå til anskaffelse av egnet RFID leser og RFID brikker, og hvordan utstyret fungerer. Videre tenker vi å se på erfaringer fra andre studier for å få den informasjonen vi trenger.

Demonstratoren vi skal utvikle skal ikke være en fullverdig implementering, men skal demonstrere noen av funksjonene. Demonstratoren skal heller ikke kobles til eksisterende systemer hos Agder Renovasjon, men vi vil i stedet beskrive forslag til hvordan dette kan ordnes. Dersom det er tid vil vi i stedet implementere en sammenkobling mot et simulert eksternt system.

Testingen av demonstratoren kommer ikke til å skje i stor skala. Avtalen med Agder Renovasjon er her at vi skal teste demonstratoren på en rute, eller et byggefelt. Dersom det mot formodning ikke blir mulig å få tak i RFID utstyr skal det likevel være mulig å implementere demonstratoren for lagring og håndtering av metadata. Vi vil i så fall lage et program som simulerer skanning av avfallsdunker.

1.6 Rapportens oppbygning

Kapittel to gir en beskrivelse av Agder Renovasjon og hvordan situasjonen deres er i dag. Agder Renovasjon er selskapet vi skal lage en testløsning for. I kapittel tre gir vi en innføring i RDF, som er teknologien vi skal teste ut som lagringsformat. I tillegg til innføringen, går vi gjennom ulike alternativ til RDF verktøy. Et av disse verktøyene blir valgt til utviklingen av en demonstrator. Kapittel fire beskriver behovet vi har til RFID utstyr, og går gjennom ulike typer lesere. Vi gjør også et valg av leser og tagger til bruk i demonstratoren.

Kapittel fem er et metodekapittel med blant annet forskningsmetode og antagelser. Vi går også gjennom ulike parametere som ligger til grunn for valg av utstyr og teknologi. I kapittel seks gir vi en kort forklaring til hva vi mener med kvalitetssikring i denne oppgaven. I tillegg tar vi for oss hvordan et system tilsvarende det vi utvikler som en demonstrator kan forbedre kvalitetssikringen hos Agder Renovasjon.

I kapittel sju kommer vi med to forslag til system. Det ene av disse er PDA basert, mens det andre er PC basert. Kapittel åtte beskriver implementasjonen av demonstratoren.

Rapporten avsluttes med resultater, drøftinger og konklusjon av demonstratoren og teknologiene vi har benyttet.

2 Agder Renovasjon – beskrivelse av case

Agder Renovasjon er et interkommunalt selskap som eies av kommunene Arendal, Froland og Grimstad. Kundene er både privatkunder og bedriftskunder. Selskapet har som oppgave å ivareta eierkommunenes renovasjonsoppgaver. Totalt har Agder Renovasjon 25300 abonnenter, og håndterer både privatkunder og bedriftskunder. 95 % av kjøringen utføres av eksterne på kontrakt via anbud. Kontraktene for kjøringen ligger på 70-80 millioner kroner per år, og gjelder fem års perioder. En ny anbudsrunde åpnet 14. januar 2005, og i september begynner 5 års perioden for dette anbudet. Agder Renovasjon er godt fornøyd med de sjåførene de har nå og håper at det blir de som får fortsette.

Selskapet har få klager i forhold til kundemassen. I fjor var det ca 20 klager per 1000 abonnenter, totalt 570 berettigede klager. Agder Renovasjon har et ønske om å forbedre dette, og er derfor med i denne oppgaven.

Vi begynte med at vi ville se på muligheter med RDF og RFID. SMARTsolutions tok kontakt med Agder Renovasjon, og vi hadde et møte med daglig leder i oktober 2004. Han var interessert i å øke kvaliteten på tjenestene til Agder Renovasjon og ble derfor med. God kommunikasjon utad er også viktig for å gi publikum et bra inntrykk av at selskapet satser på å øke kvaliteten på tjenestene sine.

Fredag 7. januar gikk vi i gang med et ”kick off” møte med ledelsen i Agder Renovasjon, Trond Kjetil Nilsen fra SMARTsolutions og David Norheim fra @semantics. Målet med møtet var å finne ut det vi trengte om hverandre, slik av oppgaven kunne gjennomføres på en best mulig måte. Vi hadde en del spørsmål til Agder Renovasjon for at vi skulle forstå mest mulig av dynamikken i selskapet, og de ville ha informasjon om hvilke tanker vi hadde rundt prosjektet.

Vi hadde blant annet disse spørsmålene til Agder Renovasjon:

- Hvordan er situasjonen i Agder Renovasjon nå?
- Hvilket utstyr eksisterer, og hvordan er mulighetene for integrasjon med dette utstyret?
- Hvilke opplysninger registreres om tømmingen nå?
- Er rutene for kjøringen fast, eller er det mye endringer?
- Hvilken rute skal demonstratoren testes på?

2.1 Dagens situasjon

Agder Renovasjon mistenker ikke sjåførene for å bevisst glemme, men sier at det som oftest er forglemmelser som fører til klager. Alle klager som kommer på tømmingen blir logget. I følge ledelsen i Agder Renovasjon så øker antall klager i ferier når det er vikarer og når det er nye sjåfører. Dersom klagen er berettiget må sjåførene rette opp feilen i løpet av neste dag. Kostnaden med å rette opp en feil er tilnærmet 400 kroner. Denne kostnaden må selskapet som har anbudet selv dekke.

Når det kommer en klage på tømming av papirdunker kan det i mange tilfeller være vanskelig å forstå hvilken dunk det er snakk om. Papirdunkene blir delt av mange, og er derfor ofte plassert på posisjonerer uten gateadresse. På grunn av disse problemene har Agder Renovasjon planlagt å registrere GPS posisjonene til alle papirdunkene ved hjelp av

Gemini utforsker. Denne registreringen skal forhåpentligvis gjøre arbeidet med å finne ut hvilke dunker kundene mener lettere.

I tillegg til vanlig avfallshåndtering har Agder Renovasjon en stor containerflåte. Det meste av kjøringen av denne utføres også av andre. Selskapet har lite kontroll på hvor containerne er, og hvordan de blir brukt. Hvert år repareres containere for 100-200000 kroner. Det er derfor ønskelig å få mer kontroll på bruken av disse. Hver gang en bil kjører inn eller ut fra Heftingsdalen passerer den vekten. Det bør derfor være relativt greit å lage et system for å registrere trafikken forbi dette punktet.

På møtet den 7. januar ble vi enige med Agder Renovasjon om at vi skulle bruke papirdunker i Grimstad til testingen av demonstratoren. I Grimstad er det plassert ut totalt 1744 papirdunker. Testingen av demonstratoren kommer derfor bare til å skje på en liten del av disse. Et mål med systemet er å kontrollere at avtalt arbeid blir utført. Det er mistanke om at noen sjåfører spekulerer i at papirdunker i grisgrendte områder ikke blir fylt opp, og derfor ikke tømmer disse så ofte som de skal. Dokumentering av tømmingen kan i så fall være med på å få disse til å utføre arbeidet i henhold til gjeldende avtaler. Gjennomføringen skal være mest mulig slik at renovatøren opplever det som en hjelp til å utføre arbeidet rett, og ikke at de føler det som en overvåkning. Målet er ikke at noen skal tas for slurv, men målet er å øke kvaliteten.

Alt avfall som kjøres inn til Heftingsdalen veies. Vekt, type avfall og liknende registreres. Noen av bilene har i tillegg vekt, slik at mengden avfall i hver dunk kan veies. Denne funksjonen blir ikke benyttet i noen stor grad nå.

3 RDF - Resource Description Framework

RDF er en spesifikasjon for en modell til å representere metadata. Metadataene beskriver ressurser, og RDF blir brukt til å koble ressursene mot hverandre og mot egenskaper. Metadata betyr data om data.

Utgangspunktet for denne oppgaven er å benytte rammeverket til å strukturere informasjon om ressurser i forbindelse med kvalitetssikring hos Agder renovasjon. Dette som alternativ til mer tradisjonelle databaseløsninger.

Vi vil i dette kapittelet starte med å gi en innføring i RDF. Deretter vil vi se på ulike verktøy som kan brukes til behandling av RDF data. Til slutt vil vi velge et av disse verktøyene til bruk i en demonstrator.

3.1 Introduksjon

RDF er et format som blir spesifisert av W3C. Første utkast til RDF ble utgitt i oktober 1997 [7]. Dette er omtrent ett år etter at første XML utkast ble utgitt. Likevel har XML hatt en mye større utbredelse enn det RDF har hatt til nå. Etter at uttrykket ”The Semantic Web” ble lansert av Tim Berners-Lee i 1999 [8], har derimot RDF blitt mer og mer aktuelt. Grunnen til dette er at Semantic Web, også kalt Internet 2, baserer seg på RDF [9]. Måten RDF er bygget på, gjør formatet lesbart for maskiner. En kobling fra en ressurs til en annen, blir beskrevet ved hjelp av en entydig egenskap. Både egenskapene og ressursene har unike identifikatorer.

Maskin til maskin kommunikasjon er i sterk vekst, og kommer til å vokse mye de neste årene. For at maskiner skal kunne kommunisere med hverandre er det derfor viktig med et felles grensesnitt for å minimalisere integreringskostnadene. RDF er et standardisert format for nettopp dette.

I en tradisjonell database, finnes det mange koblinger mellom ulike ressurser. Så snart denne informasjonen er hentet ut og for eksempel presentert på en webside, forsvinner koblingen mellom dataene. Utgangspunktet for RDF er å bevare semantikken i informasjon også etter at den er hentet ut.

3.2 Innføring i RDF

3.2.1 Grunnleggende begreper

Ressurs (subjekt)
En ressurs eller subjekt i RDF kan være hva som helst som det skal knyttes informasjon til. Det kan for eksempel være en webside, en bok eller en person. Ressursen må kunne identifiseres unikt.

Litteral
En litteral er en tekststreng som kan kobles til en ressurs. For eksempel navn på en person eller tittel på en bok.

Objekt
Et objekt er en ressurs eller en litteral som er koblet til en ressurs. For eksempel kan litteralen ”En god historie” være koblet til boka ISBN12345678.

Predikat (egenskap)
Predikat - eller egenskap - er koblingen mellom en ressurs og et objekt. Eksempler på egenskaper kan være tittel, fornavn eller ”eies av”.

Statement
Et utsagn som består av et objekt med en gitt egenskap koblet til en ressurs kalles et statement.

Tripler
Hovedidèen med RDF er å beskrive ressurser og egenskaper ved hjelp av såkalte tripler. Tripler er basisen for RDF og består av – i denne rekkefølgen – en ressurs, et predikat og et objekt. Den konstante og generelle måten å definere tripler på, gjør formatet forståelig for både mennesker og maskiner. Et eksempel på en slik triple kan være ”bilen eies av Ola”. Her er ”bilen” en ressurs, ”eies av” en egenskap og ”Ola” en ressurs. For at setningen (også kalt statement) skal kunne tolkes entydig, må den presiseres mer. Resultatet blir da for eksempel ”SR11223 EiesAv 01028012345”. Med slik presisjon må vi fremdeles anta at det første tallet er registreringsnummeret til en norsk bil, at det siste tallet er personnummer til en norsk person og at EiesAv betyr at personen eier bilen.

URI
En URI i RDF blir kalt RDF URI referanse. Dette er en unik referanse som beskriver ressursen. En epost adresse kan for eksempel beskrives som ”mailto://navn@domene.no”, mens egenskapen ”forelder til” kan beskrives som:

”http://purl.org/vocab/relationship/parentOf”

Ved at alle ressurser får en entydig definisjon på denne måten, vet vi nøyaktig hva slags ressurs som er beskrevet. Ofte finnes en tekstlig beskrivelse av egenskapene på en webside tilsvarende URIen, gitt at det er en HTTP URL.

Hvis vi fortsetter eksempelet fra triplene, kan statementet presiseres ytterligere ved å gi hvert ledd en URI. Resultatet kan da bli:

”http://www.motorvognregister.no/regnr/SR11223” ”http://vocab.no#EiesAv”
”http://www.folkeregister.no/pnr/01028012345”

Her kan for eksempel egenskapen ”EiesAv” beskrives på ”http://vocab.no”.

I dette tilfellet er det 100% entydig hva som menes. I tradisjonelle databaser, har nøklene kun betydning internt i databasen. I RDF har URIen betydning også etter at den er hentet ut.

Blank node
Et problem som kan oppstå, er når flere sammenhengende egenskaper skal knyttes til èn ressurs. For eksempel når en adresse skal kobles til en person.

Ola har adresse { 1234 Stedsnavn
Adresseveien 1

Et alternativ vil være å lage en ny ressurs, kalt OlesAdresse. Resultatet blir da:

Ole har adresse OlesAdresse
OlesAdresse har PostNummer 1234
OlesAdresse har Sted Stedsnavn
OlesAdresse har GateAdresse Adresseveien 1

Dette er også noe som er innebygget i RDF, og kalles da ”blank node” (tidligere ”anonymous resources”). En ”blank node” har ikke en URIref men blir identifisert med _:*navn*. Dersom en blank node skal være tilgjengelig utenfor grafen, må den gis en URI.

Reifisering
Et begrep i RDF er reifisering. Dette betyr ”tingliggjøring”. Et statement kan reifiseres ved å lage et nytt statement om det opprinnelige statementet. Et abstrakt begrep, for eksempel et tidspunkt, kan reifiseres til et konkret objekt i RDF som forklarer hva tidspunktet betyr. For eksempel kan ”OlesAdresse” i eksempelet for Blank node reifiseres til en Adresse.

Namespace
For å forenkle modeller, og gjøre dem mer oversiktlige, benyttes namespace som erstatning for lange URIer. Et eksempel er RDF egenskapen type, som med full URI kalles <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>. Ved å innføre et prefix ”rdf” for den største delen av URIen (http://www.w3.org/1999/02/22-rdf-syntax-ns#), kan egenskapen skrives som rdf:type. I et RDF/XML dokument (se kapittel 3.2.2) blir namespacene definert i starten.

RDF kontainere
RDF har støtte for tre ulike kontainere. ”Bag” er kontaineren der rekkefølgen av innholdet ikke har betydning. ”Alt” eller ”Alternative” er kontaineren der innholdet er likeverdig, og en av dem vil bli valgt. Et eksempel er når det er flere alternative servere som kan benyttes. Den siste kontaineren er ”Seq” eller ”Sequence” der innholdet har en gitt rekkefølge.

I eksempelet under blir bruken av sekvens demonstrert. Som prefix for <http://www.w3.org/1999/02/22-rdf-syntax-ns#> bruker vi ”rdf”.

Alle medlemmer i en kontainer får en egenskap rdf:_n der n er et tall større enn null.

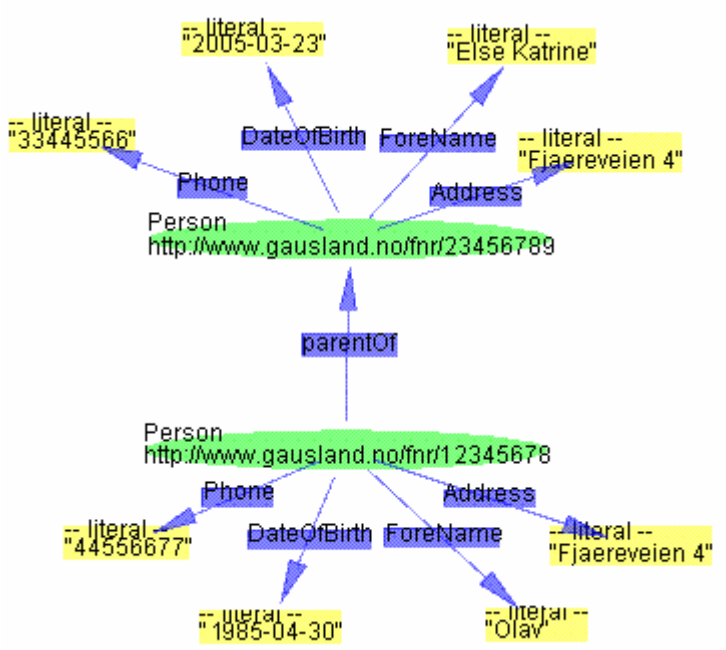
<rfid://1234/emptyed> rdf:type rdf:Seq
<rfid://1234/emptyed> rdf:_1 <rfid://1234/2005-03-30>
<rfid://1234/emptyed> rdf:_2 <rfid://1234/2005-03-31>
<rfid://1234/emptyed> rdf:_3 <rfid://1234/2005-04-02>
<rfid://1234> <http://vokab/emptyed> <rfid://1234/emptyed>

Her ser vi at en ressurs med URI <rfid://1234/emptyed> er av typen rdf:Seq. Etterpå ser vi at sekvensen har tre datoer, der rekkefølgen er bestemt av egenskapen rdf:_n. Til slutt ser vi at ressursen <rfid://1234> har egenskapen tømt, der objektet er sekvensen av datoer.

Graf

En RDF modell kan representeres som en graf. Ressurser blir angitt som ellipser, mens litteraler blir angitt som rektangler. Egenskaper blir angitt som en pil, med pilspissen mot objektet. URIen til ressursene står inni ellipsen, mens URIen til egenskapene står i nærheten av pilen.

Her er et eksempel på en RDF modell presentert som en RDF Graf. Modellen består av to ressurser, den den ene er forelder til den andre. Hver av ressursene har egenskaper som telefonnummer, fødselsdato og så videre.



Figur 2 - Eksempel på RDF Graf

Vokabular og RDFS (RDF Skjema)

Mye av det som blir representert ved hjelp av RDF har de samme egenskapene. Ett sett av egenskaper innenfor samme tema samles i såkalte vokabular. En oversikt over en god del slike vokabular finnes på [10]. Her blir det stadig lagt til nye vokabular, og eksisterende vokabular blir utvidet. Et mye brukt vokabular er ”Dublin Core Element Set” som for eksempel brukes for å beskrive språk, dato eller skaper. Et annet eksempel er vokabularet ”Relationship” som brukes for å beskrive forhold mellom mennesker (for eksempel friendOf, parentOf, siblingOf, osv). Et vokabular kan beskrives ved hjelp av elementene: Class, Property, Sub-Class, Sub-Property, Domain, Range, Comments og labels. Når vokabularet blir beskrevet av disse elementene, kalles det et RDF skjema eller et RDFS.

Fordelen med et vokabular definert som et RDF skjema, er at det da kan leses av en ”RDF parser”, og kan lett importeres i et program for å opprette RDF modeller.

Dersom det ikke finnes noe passende vokabular, er det ikke noe i veien for å definere et selv. Dersom et slikt vokabular skal gjøres tilgjengelig for andre, bør det legges på en webserver, med en katalogstruktur som ikke endres. En anbefalt måte å gjøre dette på, er å benytte årstall og eventuelt måned for tidspunktet vokabularet ble offentliggjort – for eksempel ”http://vocab.no/2005/04”. På denne måten kan vokabular oppdateres, uten at det får konsekvenser for andre som har benyttet dette, ved å legge nye versjoner i nye mapper.

Lagring
Det finnes mange ulike løsninger for å lagre RDF data. Disse bygger ofte på databaser i bunn, som MySQL eller PostgreSQL. I tillegg er det ofte mulig å lagre data ved hjelp av filer oppbygget som RDF/XML (Se kapittel 3.2.2).

Noen eksempler på program for behandling av RDF data er: Jena, Sesame, RDFStore, Kowari, 3Store og Redland. Noen av disse programmene kommer vi nærmere inn på i kapittel 3.3.

Spørringsspråk
Etter hvert har det utviklet seg flere språk som brukes til spørringer i RDF. Blant mange finnes RDQL, SPARQL og SeRQL. De fleste er bygget opp med en syntax som ligner SQL. Spørringene returnerer tripler i ett eller ett annet format. Ved hjelp av spørringer kan for eksempel all informasjon om en gitt ressurs hentes ut.

Semantikk
En stor fordel med RDF er at grammatikken, eller sammenhengen mellom dataene, er bevart også etter at de er hentet ut fra RDF database eller RDF/XML fil. Resultatet fra en spørring er en eller flere tripler. Hver for seg er disse triplene er 100% entydige ved at URIene er bevart og at triplet inneholder subjekt, predikat og objekt.

3.2.2 Lagringsformat

RDF/XML
RDF/XML er et filformat som ligner XML i oppbygning. Et eksempel på dette formatet blir vist i tabellen under.

```
<rdf:RDF
    xmlns:rel="http://purl.org/vocab/relationship/"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
  <rdf:Description rdf:about="http://family/dotty">
    <rel:parentOf rdf:resource="http://family/fran"/>
    <rel:spouseOf rdf:resource="http://family/adam"/>
  </rdf:Description>
</rdf:RDF>
```

Figur 3 - RDF/XML formatet

Som en kan se i tabellen, blir det definert to ulike namespace. Disse får en referanse (rel og rdf) som blir benyttet i resten av beskrivelsene. Alternativet ville vært å benytte hele URIen, for eksempel ”http://www.w3.org/1999/02/22-rdf-syntax-ns#Description” eller ”http://purl.org/vocab/relationship/spouseOf”. Eksempelet beskriver at ressursen med URI ”http://family/dotty” er ”spouseOf” ressursen ”http://family/adam” og ”parentOf” ressursen ”http://family/fran”. RDF/XML formatet kan leses av maskiner, og gir en entydig betydning av hva som for eksempel menes med ”spouseOf”. Definisjonen av egenskapen ”spouseOf” er ”A person who is married to this person” og er tilgjengelig på websiden tilsvarende URIen ”http://purl.org/vocab/relationship/spouseOf”.

Notation3 (N3)
Notation3 er et generelt filformat for RDF, konstruert med tanke på å bli lest av mennesker [11].

```
@prefix rel:     <http://purl.org/x/> .
@prefix rdf:     <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix :        <#> .

<http://family/dotty>
      rel:parentOf <http://family/fran> ;
      rel:spouseOf <http://family/adam> .
```

Figur 4 - Notation3 formatet

På tilsvarende måte som RDF/XML blir også her namespace definert. Dette gjør formatet mer lesbart. Hver av ressursene blir listet opp. De tilknyttede ressursene blir umiddelbart etter listet opp med type kobling og URI.

N-triples (Nt)
N-triples er et subset av N3. Formatet er linjebasert, der hver linje representerer et triple bestående av subjekt, predikat og objekt. URIene må skrives fullt ut, og elementene skilles av et eller flere mellomrom eller tab. Et triple avsluttes med punktum. [12] [13]

```
<http://family/dotty> <http://purl.org/x/spouseOf> <http://family/adam> .
<http://family/dotty> <http://purl.org/x/parentOf> <http://family/fran> .
```

Figur 5 - N-triples formatet

3.2.3 Spørringer i RDF

RQL og SeRQL
RQL er et spørringsspråk for RDF som ble utviklet av ICS-FORTH [14].

Språket ble tidigere blant annet benyttet i Sesame, som er en RDF database basert på åpen kildekode. Etterhvert har utviklerene av Sesame gått over til å utvikle et eget språk som baserer seg på RQL, RDQL, N-triples og N3. Dette språket kalles SeRQL (Sesame RQL) [15].

Et eksempel på en spørring skrevet i Sesame kan se slik ut: (hentet fra [16])

```
select
      Painter, Painting, Technique
from
      {Painter} rdf:type {cult:Painter};
            cult:paints {Painting}
            cult:technique {Technique}
using namespace
      cult = <http://www.icom.com/schema.rdf#>
```

Resultatet av spørringen er en liste over alle malere med maleriene de har laget, og hvilken teknikk de brukte.

Painter	Painting	Technique
picasso.htm	http://www.european-history.com/jpg/guernica03.jpg	"oil on canvas"@en
picasso.htm	http://www.museum.es/woman.qti	"oil on canvas"@en
rembrandt.htm	http://www.artchive.com/rembrandt/abraham.jpg	"oil on canvas"@en
rembrandt.htm	http://www.artchive.com/rembrandt/artist_at_his_easel.jpg	"oil on canvas"@en
goya.htm	http://192.41.13.240/artchive/graphics/saturn_zoom1.jpg	"wall painting (oil)"@en

RDQL
RDQL [17] er et annet spørringsspråk, som blir utviklet av HP Labs Semantic Web Research. Språket benyttes blant annet i Jena, Sesame, 3Store, RDFStore og Redland.

Språket baserer seg på SquishQL, som igjen ble basert på rdfDB. SquishQL ble brukt i databasen Inkling som ikke lenger blir utviklet.

Når eksempelspørringen over (SeRQL) blir oversatt til RDQL, vil den se slik ut:

```
select
      ?Painter, ?Painting, ?Technique
where
      (?Painter rdf:type cult:Painter),
      (?Painter cult:paints ?Painting),
      (?Painting cult:technique ?Technique)
USING
      cult FOR <http://www.icom.com/schema.rdf#>,
      rdf FOR <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

RDFQL
RDFQL [18] er et språk som benyttes i RDF Gateway fra Intellidimension. Språket ligner RDQL men er mer knyttet opp mot SQL, med for eksempel INSERT og DELETE statements.

SPARQL
BRQL, som er basert på RDQL, var det tidligere navnet på dette språket. I september 2004 ble navnet endret til SPARQL [19]. SPARQL står for Simple Protocol and RDF Query Language og spesifiseres av W3C Data Access Working Group [20].

Ifølge [21] er det ikke mange applikasjoner som implementerer SPARQL.

3.2.4 Sikkerhet

Når data blir gjort tilgjengelige som modeller på web, blir de i utgangspunktet gjort tilgjengelige for alle. Det vil også være naturlig i forbindelse med Semantic Web. I vårt tilfelle derimot, vil det være naturlig at modellene er begrenset til kun de som skal ha tilgang til informasjonen. Som vi skal se i neste kapittel, har noen av RDF verktøyene støtte for å begrense tilgang med brukernavn og passord. Alternativer til dette vil være å la serverene med databasene stå i et lukket nettverk, benytte brannmur for å kun gi tilgang til enkelte maskiner eller benytte VPN kobling mellom serverene og klient applikasjonene. Et alternativ er også å gjøre modellene tilgjengelig via sikre WebServices.

3.3 Vurdering av ulike RDF verktøy

3.3.1 Gjennomgang av ulike RDF verktøy

Det finnes etterhvert en god del verktøy til behandling av RDF. De vi har sett på er basert på åpen kildekode. Det er mange aspekter som må vurderes før et endelig valg om verktøy tas. Det vi har sett på er hvilket operativsystem som trengs, hvilket programmeringsspråk som benyttes, hva slags RDF format som kan benyttes for å hente ut og legge inn data, hvilke databaser som kan benyttes i bunn, hvilke muligheter verktøyet gir og eventuelle vanskeligheter som kan oppstå.

3.3.2 Drive

I utgangspunktet hadde vi et ønske om å programmere ved hjelp av .NET og C#. Dette på grunn av at kommunikasjon med Pocket PC, som også kan benytte C#, ville gå enklere. I tillegg har vi benyttet C# i mange sammenhenger tidligere, og anser dette som et godt språk å programmere i.

Drive [22] er et RDF verktøy som benytter C#. Programmet er greit å installere, men ikke like enkelt å bruke. Vi fikk ikke til å hente ut data som RDF/XML fil. På grunn av uoversiktlig og mangelful API og dokumentasjon, fikk vi raskt et dårlig inntrykk av programmet. Vi bestemte oss derfor for å droppe videre utforskning av Drive.

3.3.3 Jena

Jena [23] er et RDF verktøy utviklet av HP Labs Semantic Web Programme. Det er et godt RDF verktøy med støtte for å lese og skrive til RDF/XML, N3 og Nt. Som database i bunn kan MySQL, PostgreSQL eller Oracle benyttes. Jena er utviklet i Java, og er derfor platform uavhengig.

For å koble til en Jena database, må database URL, brukernavn, passord og databasetype oppgis.

```
DBConnection conn = new DBConnection(strURL, strUser, strPassword, strType);
Model m = ModelFactory.createModelRDBMaker(conn).openModel(strModel);
```

Dersom modellen ikke finnes fra før av, vil denne bli opprettet. Jena oppretter i så fall de nødvendige tabellene i databasen. Blant annet blir det for hver modell opprettet to tabeller. Den ene av disse inneholder statements, mens den andre inneholder reifiseringer. Jena oppretter også en tabell som inneholder en BLOB for hver modell. I databasen ligger alle statement med full URI. I tillegg blir det angitt om det er en ressurs eller en litteral som blir beskrevet. Et eksempel på noen rader i databasen kan se slik ut:

Subj	Prop	Obj	GraphID
Uv::http://ex.no/Person/1:	Uv::http://ex.no/05/03/person#Name:	Lv:0::Ola:	1
Uv::http://ex.no/Person/1:	Uv::http://ex.no/05/03/person#Age:	Lv:0::24:	1
Uv::http://ex.no/Person/2:	Uv::http://ex.no/05/03/person#Name:	Lv:0::Kari:	1

Jena har også en mulighet for å komprimere URIene. Dette må gjøres før databasen blir formatert for Jena. Ved å angi en største grense for lengde på URIer, vil alle prefixer til URIer som er lengre enn den angitte grensen bli lagt i en egen tabell. Deretter vil indexene til disse bli benyttet i tabellen for statements.

Model objektet har metoder for å opprette nye ressurser, litterale verdier, kontainere og så videre. Metoden ”createResource” fungerer på samme måte som metoden ”getResource”. Det vil si at om du prøver å hente ut en ressurs som ikke finnes, vil denne bli opprettet. Jena støtter RDF kontainerene ”sequence”, ”bag” og ”alt”. Så snart en metode for å opprette eller hente en ressurs eller en egenskap blir kjørt, vil Jena sørge for å opprette disse i databasen – så sant de ikke finnes fra før av. På denne måten trenger du i Jena kun å konsentrere deg om modellen, så vil resten gå av seg selv.

Modeller kan hentes inn fra filer, andre databaser eller opprettes manuelt. I Jena er det veldig enkelt å slå sammen to modeller. Dette kan gjøres ved hjelp av union, intersection eller difference.

Spørringsspråket som benyttes i Jena, er RDQL. Ifølge dokumentasjonen, bør alle spørringer utføres med read lock. Dette gjøres ved hjelp av metodene enterCriticalSection(boolean) og leaveCriticalSection(), der boolean angir om modellen skal låses for lesing eller skriving.

3.3.4 Joseki

Utviklerene av Jena har også laget et webgrensesnitt mot Jena, kalt Joseki [24]. Joseki kjøres som en egen webserver, og inneholder Jena pakken. Ved hjelp av en konfigurasjonsfil mappes modellene i databasene til en url.

Spørringer mot modellene kan kjøres som ”HTTP GET” eller ”HTTP POST”. Ved ”HTTP GET” sendes spørringen som en querystring der også spørringsspråket er angitt. Dette kan føre til problemer dersom spørringen blir større enn tillatt lengde på URL. Joseki kan også benyttes til å legge til nye eller fjerne deler av modellen.

Et eksempel på en ”HTTP GET” spørring som henter ut alle opplysninger fra en modell, kan se slik ut:

”http://renovasjon.mine.nu:2020/AgderRenovasjon?lang=RDQL
&query=SELECT * WHERE (?x ?y ?z)”

Her er ”AgderRenovasjon” referanse til modell og database i konfigurasjonsfilen. ”lang” angir spørringsspråket og ”query” angir spørringen. Konfigurasjonsfilen vil for modellen ”AgderRenovasjon” se slik ut:

```
<http://server/AgderRenovasjon>
 a                          joseki:AttachedModel ;
 joseki:attachedModel       <jdbc:mysql://url/db?autoReconnect=true> ;
 joseki:user                "username" ;
 joseki:password            "password" ;
 joseki:dbModelName         "AgderRenovasjon" ;
 joseki:dbType              "MySQL" ;
 joseki:dbDriver            "com.mysql.jdbc.Driver" ;
 joseki:hasQueryOperation   joseki:BindingRDQL ;
 joseki:hasOperation        joseki:BindingPing ;
 joseki:hasOperation        joseki:BindingQueryModel ;
 joseki:hasOperation        joseki:BindingAdd ;
 joseki:hasOperation        joseki:BindingRemove ;
.
```

Figur 6 - Del av konfigurasjonsfil for Joseki

Første linje angir navnet som blir brukt i URLen ved uthenting av data. De neste linjene angir URL, brukernavn, passord, modellnavn og databasetype. Videre kommer hvilken database driver som blir brukt, og hvilke spørringsspråk og operasjoner som skal være tilgjengelige.

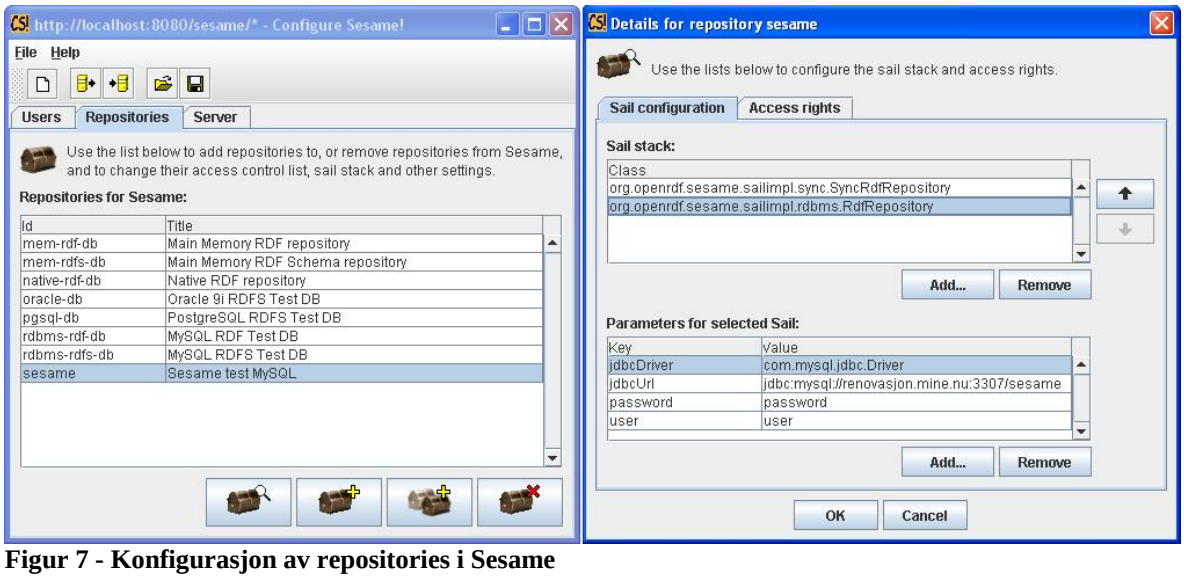
Joseki var noe tungvint å starte opp, siden det måtte startes ved hjelp av kommandovindu. Dette løste vi ved å lage en batch fil. I tillegg har vi laget en windows service som starter batchfilen ved oppstart av serveren. På denne måten er det ikke nødvendig å logge seg på med en bruker for å starte Joseki. Vi har ikke gjort noen særlig bruk av Joseki, men har fått testet å sende spørringer via nettleseren. I retur har vi fått en RDF/XML fil i med resultatet fra spørringen.

3.3.5 Sesame

Sesame [25] ble i utgangspunktet utviklet av Aduna [26] under navnet Aidministrator, som en demonstrator for EU prosjektet ”On-To-Knownledge” [27]**.** Nå er i tillegg blant andre NLnet Foundation [28] og Ontotext Lab [29] med på videreutviklingen. Sesame er utviklet i Java, og er dermed platformuavhengig.

På samme måte som Jena, har Sesame støtte for de vanlige formatene RDF/XML, N-Triples og N3. Databaser i bunn for systemet kan være PostgreSQL, MySQL, Oracle eller MSSQL. Sesame lagrer modellene i databasene på en plassbesparende måte, ved å gi alle URIene en ID. På denne måten blir ikke lange URIer gjentatt for hver rad.

For å kunne kjøre Sesame, måtte TomCat installeres. Installasjonen gikk greit. En stor fordel med Sesame, er muligheten for å begrense tilgangen til modellene. Konfigureringen skjer med hjelp av et eget konfigureringsverktøy. Verktøyet gir mulighet til å gjøre tilgjengelig modeller i ulike repository, med lese eller skriverettighet til de ulike brukerene.



Figur 7 - Konfigurasjon av repositories i Sesame

For å kommunisere med modellene, kan man velge enten HTTP eller bruke APIen. Innlogging med HTTP fungerer ved at man først sender en POST eller GET med brukernavn og passord. I retur får man en Cookie som må brukes i de resterende forespørslene mot modellen. Også med APIen refererer man til modellen med en HTTP

URL, men istedetfor å sende passord med POST eller GET, benytter man et objekt med en login metode. Brukerene kan gis både lesetilgang og skrivetilgang til de ulike modellene.

3.3.6 3Store

3Store [30], også kalt Threestore, blir utviklet av universitetet i Southampton. Programmet er en RDF Triplestore, og kan kjøres på Linux. Når vi prøvde å installere programmet under Mandrakelinux 10.1, fikk vi problemer med at installasjonen ikke fant konfigurasjonsfilene til MySQL. Etter å ha prøvd å installere en nyere versjon av MySQL, og det fremdeles ikke fungerte, prøvde vi å endre installasjonsfilene. Da oppsto imidlertid et nytt problem. Installasjonsprogrammet spurte etter Raptor og flere andre pakker. Når den ene etter den andre pakken ble installert uten positivt resultat, bestemte vi oss for å droppe videre testing av 3Store.

3.3.7 Redland

Redland [31] blir utviklet av Dave Becket ved universitetet i Bristol, og er rammen rundt en rekke program som jobber sammen for å danne et RDF system. De ulike pakkene Redland består av er:

- Raptor for å lese og lagre RDF til og fra blant annet RDF/XML og N-Triples.
- Rasqal som tilbyr spørringsfunksjonalitet. Språkene som er støttet er RDQL og SPARQL.
- Redland rammeverket med C API
- Redland Language Bindings for API til Redland i C#, Java, Perl, PHP, Python og så videre.

Redland krever at også Raptor er installert for å kunne kjøre. Raptor derimot er et selvstendig produkt. Redland kan bruke databasene Sleepycat/Berkeley DB, MySQL 3/4, AKT Triplestore og SQLite, i tillegg til filer eller URIer.

På grunn av vanskelig installasjon av Redland, bestemte vi oss for å droppe videre testing av produktet.

3.3.8 RDFStore

RDFStore [32] er det siste programmet vi har sett på. Programmet er laget i C og Perl, og ble registrert på www.sourceforge.net i 2001. Det har vært liten aktivitet på prosjektet, og versjonsnummeret er ikke kommet lenger enn til versjon 0.5. På grunn av programmeringsspråk og versjonsnummer, bestemte vi oss for å kutte ut RDFStore.

3.3.9 RDFAuthor/IsaViz

I tillegg til RDF verktøyene vi har sett på til nå, finnes det en del grafiske programmer som kan brukes til å opprette og se på modeller grafisk. Vi har prøvd to av disse.

RDFAuthor [33] benytter Jena i bunn, og er et enkelt Java program med en kjørbar .jar fil. Programmet skal kunne lese både N-Triper og RDF/XML filer. Vi har bare prøvd RDF/XML, og det fungerer fint. Programmet har en funksjon for å sette opp modellen oversiktlig, men den fungerer ikke spesielt godt. Når modellene blir store, noe de fort blir, må de fleste ressursene og litteralene flyttes på for å få en oversikt.

IsaViz [34] startes med en batch fil, og fungerer stort sett på samme måte som RDFAuthor. Også IsaViz er basert på Jena, men krever i tillegg Graphviz [35], som er et program for visualisering av grafer.

3.3.10 Valg av RDF verktøy

Etter å ha sett på de ulike RDF verktøyene, endte vi til slutt opp med å velge mellom Jena og Sesame. De andre alternativene var enten for dårlige og ufullstendige, eller hadde tungvinte installasjonsprosedyrer. De verktøyene vi i det hele tatt fikk installert, var derfor kun Drive, Jena og Sesame. Som nevnt tidligere virket Drive ufullstendig og hadde dårlig dokumentasjon. Når det gjelder valget mellom Jena og Sesame, så vi at begge verktøyene fungerer godt på Windows, og har en enkel installasjon. Begge støtter de vanligeste formatene for RDF, og begge benytter Java. I tillegg har begge mulighet for webgrensesnitt (Jena ved hjelp av Joseki). En fordel med Sesame var muligheten for å styre tilgangen til ulike modeller på brukernivå, noe som ikke var mulig med Jena eller Joseki.

Vi hadde ikke mulighet til å prøve ut både Sesame og Jena i like stor grad, siden det ville tatt for lang tid for dette prosjektet. Vi bestemte oss derfor for å se nærmere på Jena, på tross av den manglende støtten for styring av tilgang. APIen for Jena virket dessuten mer oversiktlig og fungerte så langt veldig greit.

3.4 Oppsummering

RDF er bygd opp som et nettverk av tripler, som igjen er ressurser med tilhørende egenskaper. Alle ressurser og egenskaper blir globalt unikt identifisert. På grunn av alle koblingene på kryss og tvers, blir modellene raskt store.

Som et resultat av store modeller og koblinger mellom lange URIer, vil spørringer ta lenger tid enn med tradisjonelle databaser. Det er likevel mange fordeler med RDF. Ved at alle ressurser får en global unik URI, kan flere modeller slås sammen. Dersom en URI er den samme i to modeller, betyr dette at ressursen er den samme. En rad med ID=321 i en tradisjonell database kan ha ulike betydninger i ulike databaser, og har følgelig ingen betydning utenfor databasen.

Det finnes etterhvert mange verktøy for RDF. Noen av disse kan brukes på en fornuftig måte, mens andre fremdeles er på utviklingsstadiet. Jena og Sesame er to av de mest kjente og brukte verktøyene for RDF, og begge fungerer godt.

4 RFID utstyr

4.1 Introduksjon

Tekonologien RFID benytter radio frekvenser til å utveksle data mellom en RFID brikke og et datasystem via en kontrollenhet med en antenne. Vanlige RFID brikkene har en unik ID som gjør at de kan brukes til å unikt identifisere objektet de blir festet til. De fleste brikkene har i tillegg muligheten til å lagre noe data på brikken.

RFID teknologien kommer til å ha en stor vekst i løpet av de neste årene. Prisene på utstyr har begynt å synke mot nivå som vil føre til at mange flere tar teknologien i bruk. Siden dette ikke er en studie av RFID kommer vi ikke til å gå inn i detaljer i teknologien, men vi gir en liten oversikt over temaet.

RFID har ikke hatt en stor utbredelse i Norge ennå, men flere og flere bedrifter har startet opp prosjekter med bruk av RFID. Gilde på Gol står bak et av de større prosjektene. De har i noen år merket 90000 sau per år med RFID brikker i øret. I følge EAN Norge spares det millionbeløp på automatisk avlesning av øremerkene. Bakgrunnen for Gildes prosjekt er EUs skjerpede krav til sporbarhet for kjøtt. [36] [37]

Utstyret som finnes på markedet er veldig variert både når det gjelder egenskaper og pris. Brikkene er delt inn i ulike kategorier. De vanlige og enkleste brikkene er passive. Disse har ikke batteri, og får energi kun fra radiobølgene som leseren sender ut. De enkleste av disse har kun en ID og ikke noe minne. Semipassive brikker har batteri, men dette brukes kun til databehandling. Responsen tilbake til leseren sendes også her med energi fra radiobølgene. Den siste kategorien er aktive brikker. Disse har batteri som brukes til databehandling og sending. Rekkevidden på disse brikkene er derfor mye lenger enn på passive og semipassive brikker. Frekvensene som benyttes er fra litt over 100 kHz til 5,4 GHz.

Rekkevidden for avlesning av passive RFID brikker er liten. Stasjonære lesere kan benytte stor sendeeffekt og på denne måten få en rekkevidde på noen meter. Bærbart utstyr vil typisk ikke kunne lese brikker på større avstand enn 5 til 10 cm. Når det gjelder lesing av aktive tagger er det derimot mulig å oppnå lange avstander selv med liten sendeeffekt. I følge [38] kreves det bare 1 mW for å oppnå kommunikasjon på 100 meter ved bruk av 433 MHz system med aktive brikker.

RFID kan benyttes til merking av mange objekter som kan befinne seg i samme område. I slike tilfeller er det viktig at utstyret blant annet har antikollisjonsmekanismer. I denne oppgaven er objektene som er merket så langt fra hverandre at dette ikke er noe problem. Vi har derfor valgt å ikke gå inn på dette temaet.

4.2 Behov for utstyr til demonstrator

Agder Renovasjon hadde et ønske om å teste en demonstrator på papirdunker. Vi har derfor konsentrert oss om utstyr til nettopp det. Til implementeringen av RFID delen i denne oppgaven kunne vi ønsket oss RFID brikker som kan skrus eller nagles fast til papirdunkene. I tillegg ville vi hatt RFID lesere montert bak på bilene. Disse skulle hatt en rekkevidde på ca 60 – 80 cm slik at brikkene på dunkene automatisk ble lest når de ble

tømt. Men rekkevidden måtte ikke vært større, for da kunne dunker blitt registrert bare av å kjøre forbi.

På grunn av den korte tiden vi har til rådighet i forbindelse med oppgaven har vi valgt å satse på bærbart utstyr. På den måten sparer vi en del tid på montering av utstyret, samtidig som vi enkelt kan flytte utstyret frem og tilbake mellom skolen og renovasjonsbilen. På denne måten blir testingen enklest mulig. Ønsket var derfor en bærbar RFID leser som automatisk registrerte brikkene på dunkene når renovatøren var i nærheten. Dersom leseren ble festet i beltet eller liknende ville den være i en grei høyde i forhold til dunkene, og en rekkevidde på 50 – 60 cm ville vært passelig.

Vi gikk i gang tidlig i prosjektperioden med å finne RFID utstyr som passet vårt bruksområde, slik av vi skulle få dette på plass så tidlig som mulig. Vi så etter utstyr som var godt egent til testingen her. Utstyret må være enkelt å håndtere, og må være robust nok til å tåle behandlingen. I tillegg bør det kunne benyttes på andre områder senere, siden det her kun skal utvikles en demonstrator, og ikke et ferdig system. Med bærbart utstyr vil det også være mulig å lage løsninger for blant annet restavfall, matavfall og glass- og metallemballasje. Slik situasjonen er nå, er det kun papirdunker som trilles bort til bilene. Annet avfall blir hentet og løftet bort til bilene, slik at dunkene aldri er i umiddelbar nærhet av bilen.

Etter litt undersøkelser fant vi ut at det ikke eksisterte bærbart RFID utstyr med den rekkevidden vi ønsket. Det meste av utstyret vi fant hadde en rekkevidde på rundt fem centimeter. Denne korte rekkevidden fører til at brukeren må skanne dunkene manuelt, det nytter ikke å bare være i nærheten av dunken. For å kunne gi brukeren tilbakemeldinger dersom en dunk glemmes, eller fortelle adressen til neste dunk, må leseren være mulig å programmere, eller koblet til noe som kan programmeres.

Til demonstratoren vår har vi behov for å kunne unikt identifisere dunkene. Aktive RFID tagger koster mye mer enn passive, og inneholder batteri som må skiftes etter en viss tid. Utvalget av passive tagger er stort, og det finnes mange varianter for forskjellig bruk. Vi valgte å kontakte noen leverandører for å få tips til tagger som passet til denne casen. Taggene trenger ikke ha noe særlig minne, men må være litt robuste slik at de tåler tøff behandling i tillegg til vær og vind.

Fra SMARTsolutions fikk vi låne en iPAQ som kjører Pocket PC. Denne har CF port og innebygd Bluetooth og IR. Siden vi fikk låne denne konsentrerte vi oss om å finne utstyr som passet sammen med denne for å slippe å kjøpe inn en ny PDA.

4.3 RFID lesere

I vårt søk etter RFID lesere kom vi over mange som var fysisk store og uhåndterlige, i tillegg var mange av leserne umulige å programmere. PSI Systems informerte oss om at de hadde en RFID leser fra Symbol [39] som kjørte Pocket PC, men denne kostet mellom 30- og 35000 kroner. Siden vi da hadde fått tilgang på en iPAQ, fant vi ut at det var greiest å satse på en leser som kunne kobles denne.

4.3.1 TEK iPAQ og TEK Protégé Tungsten C

Dette er en RFID leser som blant annet kommer i utgaver for HP iPAQ 5555 og Palm Tungsten C. Denne leseren bruker 6 oppladbare NiMH batterier. Dette gjør at den ikke trekker strøm fra PDAen den er tilkoblet, men gjør også at den fysiske størrelsen og vekten blir stor. Målene er: 3,3 tommer bredde, og 7,4 tommer lengde. I centimeter blir dette ca 8,3 x 18,8. Leseren kan lese RFID brikker med frekvens: 13.56 MHz. Vi fant utgaven for Palm Tungsten C til 429 $ hos RFIDusa.com. [40] [41]

4.3.2 IDBlue

IDBlue [42] ser ut som en markeringspenn. Den har innebygd Bluetooth som den bruker til å kommunisere med, og kan kan kobles mot mye forskjellig utstyr, slik som bærbar PC, stasjonær PC, PDA og mobiltelefon. Pennen benytter Bluetooth 1.1 klasse 2 med opptil 15 meters rekkevidde. IDBlue kan også benyttes uten at den har kommunikasjon med andre. I følge websidene til produsenten, Cathexis Innovations Inc, kan det innebygde minnet brukes til lagring av ID til flere tusen RFID brikker når enheten ikke er koblet til en PDA eller liknende. Disse dataene kan sendes til en annen enhet senere. Vi har i etterkant vært i kontakt med Cathexis Innovations Inc og fått opplyst at denne funksjonen ikke er helt på plass ennå, men at dette er noe de har planer om å ordne i senere versjoner.

IDBlue blir levert med utviklingsverktøy for Visual Studio.NET 2003. Utviklingsverktøyet inneholder en versjon for Pocket PC 2003 og en for PC. Dersom det er behov kan IDBlue leveres med et beskyttende futteral med belteklips. På den måten er den både beskyttet, og brukeren unngår å miste den.

Frekvensen som leseren opererer på er 13.56 MHz. Fysisk størrelse er 112 x 25 x 16 millimeter, vekten er 50 gram og pris fra produsent er 500 $. Denne leseren håndterer ikke lesing av flere tagger samtidig, men med en rekkevidde på ca 5 cm bør det ikke bli noe stort problem.

4.3.3 Socket RFID leser for CF port

Socket Communications Inc har laget en RFID leser som benytter Compact Flash porten på en PDA med Pocket PC [43]. Leseren får strømmen den trenger fra Pocket PCen, men i følge databladet skal den trekke så lite strøm at brukeren kan benytte den i flere timer uten å måtte lade. Rekkevidden er som for annet bærbart utstyr kort. Socket oppgir denne til å være ca 6 cm.

Leseren blir levert sammen med programmet SocketScan. Dette er et program som formidler innleste data som virtuelle tastetrykk til hvilket som helst Windows program. Integrering med eksisterende applikasjoner vil derfor kunne være enkel.

Leseren leser RFID brikker på frekvensen 13.56 MHz. Størrelsen på hodet til leseren er 45 x 49 x 21 millimeter. Prisen på denne var 1995 $ i juli 2004 [44].

4.3.4 Syscan RFID leser for CF port

Syscan lanserte i 2003 en RFID leser [45] for Compact Flash port. Denne leseren skulle i følge sjef i Syscan, Axel Striefler, åpne RFID markedet for mange nye anvendelsesområder [46]. Dette skyltes at RFID lesere var svært dyre, mens denne skulle koste mindre enn 150 $.

Egenskapene til denne likner på Socket RFID leseren.
Begge leser tagger på 13.56 MHz, og begge benytter CF port til koblingen mot en PDA. Syscan har valgt å benytte CF type II, som er litt tykkere enn CF type I. Denne leseren vil derfor kun være mulig å benytte på PDAer som støtter CF type II.

Leseren opererer på 13.56 MHz, størrelsen på hodet er 45 x 35 x 12 millimeter.

4.3.5 Valg av leser

I tabellen under, sammenligner vi de ulike leserene vi har vurdert som alternativer til dette systemet. Viktige faktorer er pris, størrelse og støttede tagger.

Tabell 1 - Sammenligning av RFID lesere

	TEK iPAQ / Protègè	**IDBlue**	**Socket RFID**	**Syscan RFID**
Frekvens	13,56 MHz	13,56 MHz	13,56 MHz	13,56 MHz
Leser	Diverse merker	ISO 15693-2, ISO 14443A	ISO 15693, ISO 14443A	ISO 15693, ISO 14443
Tilkobling	Egen løsning	Bluetooth	CF port	CF port type II
Strømkilde	Egne batteri	Egne batteri	Via CF port	Via CF port
Strømforbruk passiv	Ukjent	< 10 mA	13 mA	15 mA
Strømforbruk aktiv	Ukjent	30 mA	52 mA	55 mA
Rekkevidde	Ukjent	5 cm	6 cm	5-10 cm
Mål	84 x 188 x ca 30 mm	112 x 25 x 16 mm	45 x 49 x 21 mm (hode)	45 x 35 x 12 mm (hode)
Vekt	Ukjent	50 g	34 g	Ukjent
Pris	429 $	500 $	1995 $	< 150 $

Som tidligere nevnt er rekkevidden på bærbart RFID utstyr veldig begrenset. På grunn av den korte rekkevidden må leseren nærmest helt inntil brikken som skal leses. Dette vil bli tungvint med en stor klumpete leser. Leseren fra TEK er som vist i tabellen den største av dem vi har sett på. TEK leseren for iPAQ passer sammen med iPAQ 5555, men den fikk låne var iPAQ 3970. Størrelsen, samt at vi måtte kjøpt en ny PDA førte til av vi valgte å kutte ut TEK leseren.

IDBlue fra Cathexis Innovations Inc. er betydelig mindre enn alle andre vi har sett. Størrelsen gjør at den blir veldig enkel å håndtere og enkel å ta med seg. Utformingen er ergonomisk, og har også den ekstra funksjonen at leseren kan brukes som en stylus[1]. I

[1] Penn som brukes til å skrive på skjermen på en PDA.

tillegg har den Bluetooth, slik at den kan kobles sammen med det meste av utstyr som har Bluetooth. I vårt tilfelle vil dette være en PDA med Pocket PC, men det kunne like godt vært en bærbar PC, stasjonær PC, mobiltelefon eller en tablet PC. Som tidligere nevnt har IDBlue minne slik at den skal kunne brukes uten å være tilkoblet noe annet utstyr. Dette kan være en ulempe dersom det er viktig å vite når en avlesning skjedde. Driverne til IDBlue krever versjon 2003 av Pocket PC, den vil derfor ikke uten videre kunne benyttes sammen med iPAQen vi fikk låne fra SMARTsolutions.

Leserne fra Socket og Syscan virket veldig greie, men har i likhet med TEK leseren ulempen med at det blir litt tungvint å måtte ta hele PDAen helt inntil for å lese av taggene. Dette medfører at det bør brukes en PDA som er laget spesielt for å tåle store påkjenninger og belastninger.

Av hensyn til brukerne valgte vi å benytte IDBlue til demonstratoren. Vårt inntrykk er at det er denne leseren som er mest brukervennlig. PDAen som brukes sammen med leseren kan ligge i en lomme, og er dermed straks mer beskyttet for vær, vind og ytre påkjenninger. Brukeren trenger kun å betjene en liten enhet som kan henge i belte. Problemet med at vi måtte ha Pocket PC 2003 løste vi ved å kjøpe en lisens på Pocket PC 2003, og oppgraderte iPAQen.

4.4 RFID tagger

IDBlue RFID leseren som vi bestemte oss for å bruke, kan lese tagger som følger standardene ISO 15693-2 og ISO 14443A. Den leser i tillegg tagger av typen Tag-it og den kommende typen HF/HFI. Vi kontaktet Cathexis Innovations Inc. for å høre om de hadde noen anbefalinger av tagger som kunne monteres utendørs og som tåler litt tøff behandling. Cathexis anbefalte oss å kjøpe kredittkort tagger. Disse er formet som kredittkort, og kretsene ligger godt beskyttet inni plasten. Prisen på disse taggene var høy og de måtte sendes fra Canada.

PSI Systems AS [47] tilbød oss tagger som skulle passe til bruken vår. Disse var også samme størrelse som kredittkort, men lå ikke plassert inni plast på samme måten. Taggene var laget som klistremerker, slik at de kunne klistres rett på objektene som skulle merkes. Prisen på disse var kun 1/3 i forhold til taggene fra Cathexis Innovations Inc. PSI Systems AS kunne også levere raskere, så vi bestemte oss for å bestille derfra.

5 Systemering

5.1 Type design og underliggende antagelser

Designen vi bruker her er en kvalitativ tilnærming. Et kvalitativt studie ender som regel opp med foreløpige svar og hypoteser. I dette tilfellet vil vi prøve å finne svar på om RDF kan være nyttig fremfor tradisjonelle teknologier. Våre teorier kan senere bli grunnlag for en kvantitativ studie for å bekrefte eller avkrefte disse.

Vi antar at innføring av RFID hos Agder Renovasjon vil medføre økt kvalitet på tjenestene. Dette på grunn av varsling dersom en avfallsdunk glemmes og i tillegg gir det mulighet til å dokumentere det arbeidet som blir utført.

Ved bruk av RDF vil det også bli enklere å integrere informasjon fra de ulike systemene, noe som gjør det enklere å utvikle nye tjenester. Nye tjenester kan for eksempel være kobling mot geografiske informasjonssystemer (GIS), eller fakturering basert på antall tømminger, slik som i Kristiansand kommune [48].

Under har vi satt inn en tabell som viser plasseringen av prosjektet vårt i et forskningsrammeverk. Her er det fire arbeidsmetoder; bygge, evaluere, utvikle teori og plassere. Ved hjelp av disse metodene kan vi komme fram til begrep, modeller, metoder eller produkt.

Tabell 2 - Forskningsrammeverk

	Bygge	Evaluere	Utvikle teori	Plassere
Begrep				
Modell				
Metode		X		
Produkt	X			

Som vi ser av tabellen har vi tenkt å evaluere en metode. Metoden vi har tenkt å evaluere, er bruk av RDF for lagring og håndtering av metadata. Vi har også tenkt å lage et produkt, i dette tilfellet en demonstrator, som benytter seg av metoden vi har evaluert.

5.2 Studentenes bakgrunn

Vi har bakgrunn fra 3-årig bachelor utdanning innen Datateknikk. På masterstudiet har vi tatt fordypning innen systemutvikling. Dette innebærer blant annet kunnskaper om modellering, programmering og databaser. Disse elementene er viktige i forbindelse med utvikling av demonstratoren. Vi antar også at den kunnskapen vi har om systemutvikling vil være et godt grunnlag for å kunne tilegne oss den nye kunnskapen vi trenger om RDF, RFID og EII.

5.3 Datainnsamling og databehandling

Dette studiet er et kvalitativt case studium, der vi skal studere nytten av en teknologi i et gitt case. I et case studium er det vanlig å observere, intervjue og benytte seg av hensiktsmessige dokumenter.

I dette studiet har vi blant annet tenkt å se på erfaringer fra sjåfører og ledelse, etter at en testperiode er gjennomført. Vi vil også observere underveis. Erfaringene vil vi samle ved å intervjue sjåfører og ledelse. Fra sjåfører kan vi få nyttige erfaringer om demonstratoren fungerer, og hvordan den er i bruk. Fra ledelsen kan vi finne ut om teknologien er interessant med tanke på kvalitetssikring og dokumentasjon.

Ved å observere sjåfører vil vi kunne se om utstyret blir brukt riktig og om det er brukervennlig. Vi vil også selv kunne se om tømminger blir registrert riktig. Underveis kan vi bruke disse opplysningene til å videreutvikle og forbedre systemet.

For å se effekten av teknologien, burde testperioden gått over lenger tid. Den ruten som er valgt ut, blir tømt en gang hver uke. Dette gir oss mulighet til å teste utstyret fire til fem ganger.

I tillegg til erfaringer fra ansatte i Agder Renovasjon, må vi se på resultatet fra skanningene. Vi må hente ut RDF dataene enten manuelt eller ved å lage et program som gjør dette for oss.

Det kan bli vanskelig å finne ut om RDF faktisk er til nytte i denne sammenhengen ved å intervjue sjåfører og ledelse. RDF er en teknologi som ligger i bakgrunn for systemet. Vi må derfor se på våre egne erfaringer med denne metoden å strukturere metadata på, og gjøre antagelser om hvordan RDF er i forhold til andre teknologier.

5.4 Gyldighet på innsamlede data

Det at en person vet at han blir observert kan være nok til at handlingsmønsteret blir påvirket. Personen kan for eksempel prøve å utføre et bedre arbeid enn han ville gjort ellers. For at våre observasjoner skal bli minst mulig påvirket av at vi er tilstede skal vi informere personene som deltar om at det ikke er de som blir observert, men at det er utstyret som skal observeres.

Et problem med vårt studie er at perioden er så kort. I et kvalitativt studie bruker en ofte måneder og år på observasjoner og innsamling av data. På grunn av den begrensede tiden vi har til rådighet i dette prosjektet, vil det være problem å samle inn nok data til å kunne fastslå om det har vært forbedringer i kvaliteten. Vi vil likevel kunne gjøre antagelser om bruken av RDF.

5.5 Parametere for valg av utstyr

Det er flere parametere som ligger til grunn for valg av utstyr og løsninger. Et viktig punkt for Agder Renovasjon er kvaliteten eller varigheten på utstyret. De er for eksempel ikke interessert i å kjøpe inn store mengder RFID tagger som kanskje må skiftes ut etter et år i bruk. Det er også viktig at utstyr for å lese RFID brikker med stor sannsynlighet er kompatibelt med eventuelt nye brikker som kjøpes inn. For bærbart utstyr spiller også batteritid en stor rolle.

I tillegg til kvalitet, har pris og brukervennlighet stor betydning. For at renovatøren skal ha interesse av å ta i bruk utstyret, må det være enkelt å bruke og samtidig gi noe igjen. Det er en stor fordel om utstyret fører til at sjåføren slipper å bruke tid på å kontrollere dunker det har vært klager på, noe som fører til større effektivitet.

Når det kommer til valg av lagringsteknologi, har blant annet sikkerhet, krav til lagringsplass og hastighet stor betydning. For en applikasjon som skal hente ut data og fremstille informasjonen i tabeller eller grafisk, er det viktig at det ikke tar for lang tid å få svar på ulike spørringer. Synkronisering av informasjon om ruter som skal tømmes og som har blitt tømt kan foregå utenom arbeidstid, så her spiller ikke effektivitet så stor rolle. Argument for å velge RDF er integreringsmuligheter og enkel kobling av informasjon. Argument for standard relasjonsdatabaser er plassforbruk og effektivitet.

6 Kvalitetssikring

6.1 Hva er kvalitetssikring?

Kvalitetssikring er i denne sammenhengen å forsikre seg om at arbeid som skal utføres, faktisk blir utført på en tilfredsstillende måte. Ved å ha god kvalitetssikring og gode mål for hvordan arbeidet skal utføres, sikres kvaliteten på tjenestene. Dette vil føre til bedre kundetilfredshet, men koster tid og penger. Oftest blir kvalitetssikring utført internt i bedriften. Tiltakene for kvalitetssikring kan for eksempel være krav til kompetanse, fastsetting av rutiner, kontrollrutiner og ekstern vurdering [49].

Når kvaliteten er på linje med definerte standarder i bransjen, kan bedriften få en sertifisering for sitt kvalitetssikringssystem av et uavhengig offisielt organ.

6.2 Kvalitetssikring hos Agder Renovasjon

Slik situasjonen hos Agder Renovasjon er i dag, er det lite kontroll med hva som blir tømt og når. De har en plan for tømmingen, men har ingen mulighet for å i etterkant kunne dokumentere at arbeidet er utført etter planen. Dersom en kunde klager på at avfallet ikke er hentet, må ofte Agder Renovasjon ut og kontrollere selv om de er 100% sikre på at klagen ikke er berettiget. I noen tilfeller har kunden rett, men dersom dette for eksempel gjelder en felles papirdunk, kan det være vanskelig å vite nøyaktig hvilken lokasjon klagen gjelder. Ved å innføre merking av dunker og registrere tømminger, vil dette føre til bedre dokumentasjon. Om dunkene merkes med RFID brikker eller strekkoder, er i utgangspunktet likegyldig. Det samme gjelder valget mellom vanlig SQL database eller å benytte en RDF løsning. For kvalitetssikringens del, er det viktigeste at dunker kan identifiseres og at informasjon om tømminger kan knyttes til disse.

For kvalitetssikring, er det også viktig at informasjonen som registreres beholder sin integritet. Det må ikke være mulig å manuelt manipulere informasjonen fra tømmingene. Ved å la en tredjepart ha kontroll over databasen, vil det bli vanskeligere å manipulere resultatene. Det vil fremdeles være mulig å manipulere informasjonen før den blir lagt inn i databasen. Ved å benytte GPS eller GPRS for å overføre informasjonen direkte etterhvert som den registreres, reduseres muligheten til manipulasjon til et minimum.

Et problem med bærbart utstyr, er muligheten for å skanne dunker selv om disse ikke blir tømt. Samtidig blir det tungvint å måtte dra en dunk bort til bilen for å tømme denne dersom den i utgangspunktet er tom. Det blir derfor nødvendig å måtte stole på at en dunk blir skannet kun dersom den er tom eller blir tømt. Ved å også registrere GPS posisjonen til bilen etterhvert som den kjører ruta, vil det enkelt kunne dokumenteres at bilen har vært på alle stoppene den skal og hvor lenge den har vært på hver stopp. Det er da lite sannsynlig at sjåføren ikke tømmer de dunkene han skal. Ved en eventuell klage, kan man i etterkant enkelt finne ut om jobben er utført eller ikke.

En annen måte å kunne forbedre kvaliteten på, er å varsle sjåføren så snart han har passert en dunk som skulle vært tømt. På denne måten kan feilen rettes på et så tidlig tidspunkt som mulig.

Når all relevant informasjon som hvem som tømte, hviklet tidspunkt, avfallstype og så videre blir registrert i systemet, vil Agder Renovasjon kunne dokumentere hva som er utført og samtidig kunne redusere antall feil betraktelig.

Når det gjelder valget mellom RFID og strekkoder på dunkene, er RFID stort sett å foretrekke. Ulempen er prisen både på brikker og lesere. I tillegg er det problematisk å montere RFID brikker på metall. Valget mellom RDF database med SQL database i bunn, eller kun SQL database er litt verre. Dette vil vi komme tilbake til i resultater og drøfting av RDF.

7 Forslag til system

Basert på opplysningene vi har fått fra Agder Renovasjon om dagens situasjon har vi satt opp to forskjellige forslag til hvordan et system for å bedre kvalitetssikringen kan implementeres gjennom bruk av RDF og RFID. Mulighetene for variasjon innenfor forslagene er store. Før presentasjonen av disse to forslagene presenterer vi et tenkt scenario for hvordan et slikt kvalitetssikringssystem kunne blitt implementert. Dette forslaget er et teoretisk forslag som ikke er ment som et reelt forslag til Agder Renovasjon.

7.1 Tenkt scenario

Se for deg at du har en opprydning hjemme. I den forbindelse blir det en del ekstra avfall. Du sorterer avfallet, men dunken for restavfall blir full. I følge renovasjonskalenderen er det seks dager igjen til restavfallet skal tømmes. I stedet for å bekymre deg over at dunken er full og at avfallet kommer til å hope seg opp, forsetter du heller med andre ting. Avfallsdunken varsler automatisk renovasjonsselskapet om at den er full. Neste dag når du kommer hjem fra arbeid, er dunken tømt og du slipper opphopning av avfall.

I Danmark eksperimenteres det allerede med avfallsdunker som selv varsler når de er fulle [50]. Disse dunkene registrerer også vekten på avfallet.

I renovasjonsbilene er det montert PC med berøringsskjerm på dashbordet. Bak på bilene er det montert RFID leser. RFID leseren, en GPS modul og en GPRS modul er koblet til PCen. Renovatøren får opp all nødvendig informasjon om ruten på skjermen. Ved opplæring av nye sjåfører trenger en derfor ikke å bruke tid på å lære sjåførene de ulike rutene. Sjåførene ser hele tiden et elektronisk kart som viser hvor de skal kjøre, hvilke dunker som skal tømmes og hvilke som er tømt.

Hver dunk som tømmes veies samtidig, og vekten blir registrert med informasjonen om tømmingen. Over tid får en statistikker på hvor mye avfall det vanligvis er i de ulike dunkene. Dette brukes til logistikk optimalisering, slik at en får en mest mulig effektiv kjøring. Under kjøringen sendes informasjon om tømmingene via GPRS, slik at informasjonen på serveren er mest mulig oppdatert. Ved endringer av ruten får sjåføren oversendt den nye informasjonen om ruten.

Renovasjonsavgiften baseres på hvor mye avfall kunden kaster. For å gjøre det mest mulig attraktivt å gjenvinne avfall, er renovasjonsavgiften todelt. Avgiften består av en fast del som dekker kostnadene for gjenvinnbart avfall, og en variabel del som går på hvor mye avfall en kaster som ikke kan gjenvinnes. På denne måten blir det de som kaster mest avfall som må betale mest, og de som kaster lite betaler lite. Det blir slutt på at en gammel dame betaler like mye som en stor småbarnsfamilie.

Moderniseringsminister Morten Andreas Meyer står bak borgerportalen ”Min Side”. Borgerportalen skal gjøre kommunikasjonen med det offentlige lettere, og samle all informasjon på et sted. Informasjonen om hvor mye avfall hver privat kunde har kastet, vil vi derfor integrere med resten av informasjon på ”Min Side”. På denne måten kan kundene logge seg inn for å få en oversikt over hvor mye avfall de har kastet.

I dette systemet vil vi ha en RDF database i bunnen. Dette vil bidra til å gjøre det enkelt å tilrettelegge informasjon til blant annet ”Min Side” og miljøverndepartementet.

Dette er som nevnt et tenkt scenario. Alt som er nevnt er praktisk mulig, men det vil ikke være økonomisk forsvarlig å implementere alt.

7.2 Forslag 1, PDA basert implementering

Dette forslaget baserer seg på at renovatørene utstyres med en PDA og en IDBlue enhet. PDAen skal inneholde informasjon om hvem som er sjåfør, ruten som skal kjøres, og hvilke avfallsdunker som er med på ruten. Etter hvert som dunkene blir tømt, registreres tidspunktet.

Renovatøren kan se en liste over dunkene som er med på ruten. Han eller hun kan se hvor neste dunk er dersom det er usikkerhet. Dersom en dunk blir glemt kan det gis varsel om dette, slik at det kan rettes på et så tidlig tidspunkt som mulig. Her finnes flere alternativer for hvordan varslingen blir aktivert. For ruter som er helt faste kan det gis varsling allerede ved første dunk etter en glemt dunk. Ruter med litt mer variasjon kan for eksempel deles inn i områder, slik at det gis varsel dersom en dunk i et område ikke er tømt når renovatøren er ferdig med dette området.

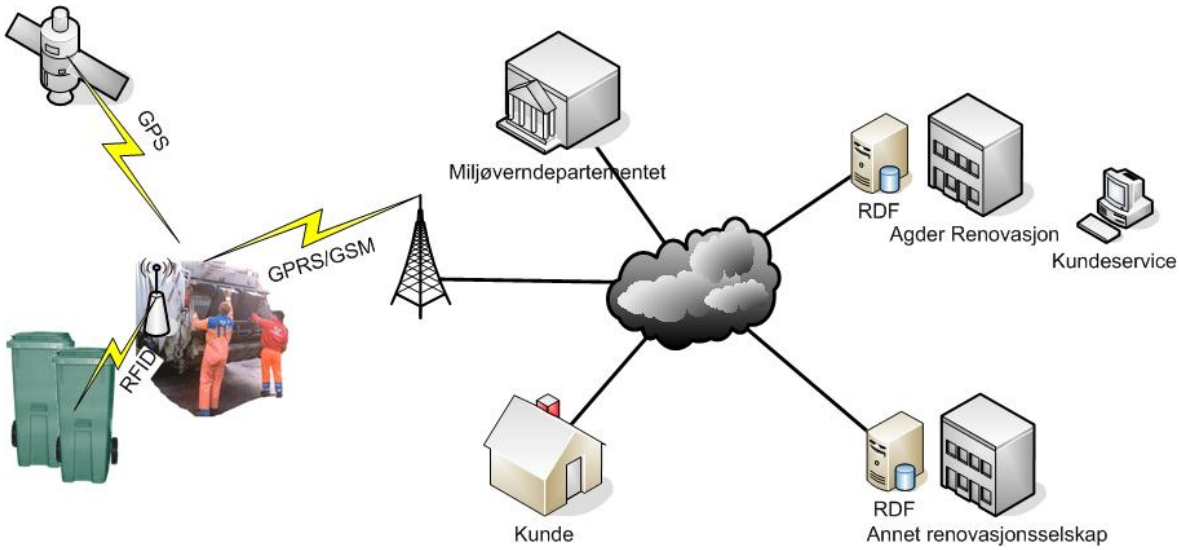
Forslaget kan benyttes på de fleste typer avfall, så lenge det kan merkes med RFID brikker. På papiravfall kan det benyttes RFID brikker på dunkene, for restavfall, matavfall, metall- og glassemballasje og lignende kan det festes RFID brikke i kildesorteringsskapet.

Den enkleste versjonen av dette forslaget benytter seg av en vanlig PDA. Dataene hentes ut fra denne, og legges inn på serveren når renovatøren kommer tilbake til Heftingsdalen. En litt mer avansert utgave vil benytte en PDA som har GPRS. På denne måten kan dataene sendes direkte til en server etter hvert som tømmingen utføres. Det vil også være mulig å benytte en GPS modul i PDAen, slik at ruteinformasjonen blir best mulig. Både bilen og dunkene kan da plottes inn på et kart som vises på skjermen. I sammenheng med GPRS vil det også være mulig å ha oversikten over hvor bilene befinner seg.

På serveren kjører en RDF database, denne brukes til å lagre alle dataene om rutene og tømmingene. Denne informasjonen utveksles mellom PDA og server når en PDA synkroniseres. Informasjonen om rutene blir lagt over på PDA, og informasjon om dunkene som er tømt legges over på serveren. Siden vi ikke har funnet noen ”RDF parser” som fungerer på Pocket PC foreslår vi at det benyttes et annet format på PDAen, og at dataene blir konvertert til RDF på serveren. Med RDF databasen i bunnen kan for eksempel miljøverndepartementet få lov til å stille spørringer for å få oversikt over hvor mye avfall som er kastet.

7.3 Forslag 2, PC basert implementering

Det andre hovedforslaget baserer seg på at det monteres PC i renovasjonsbilene. Vi foreslår at disse utstyres med GPS og GPRS modul. Med en GPS modul vil en få bilens posisjon, og denne kan plottes inn på et kart på skjermen sammen med dunkene som skal tømmes. På denne måten kan renovatøren enkelt se hvor han eller hun skal kjøre. Opplæringen av nye sjåfører kan bli veldig enkel, siden de får ruten de skal kjøre opp på skjermen i bilen. GPRS modulen sørger for kommunikasjon med sentral server, slik at oppdateringer på hvilke dunker som er tømt, og annen informasjon kan formidles.



Figur 8 - Oversiktsbilde, PC basert løsning

Vi foreslår at det differensieres på type avfall. Papiravfall som er i plastdunker trilles bort til bilene, og tømmes direkte i bilene. Restavfall, matavfall, glass- og metallemballasje hentes for hånd, og kastes i bilene. Siden det er relativt store forskjeller på dette, vil vi utstyre bilene forskjellig.

Bilene som tømmer papiravfall blir utstyrt med en fastmontert RFID leser bak på bilen. Denne leser automatisk av dunkene som tømmes, og tømmingen blir registrert. På disse bilene kan de i tillegg monteres en vekt. Dette vil gi statistikker på hvor mye avfall det er på de ulike stedene. Denne informasjonen kan over tid blant annet brukes til å avgjøre hvor det er behov for flere dunker, eller hvor det er plassert ut flere dunker enn det er behov for.

Siden restavfallet hentes og bæres bort til bilen, er aldri bilen i umiddelbar nærhet av dunkene. På grunn av dette foreslår vi å montere en RFID brikke i skapet ved dunken. Brikkens ID kobles til kunde og adresse, og den samme brikken kan brukes til både restavfall, matavfall, glass- og metallemballasje og spesialavfall. Bilene som tømmer dette avfallet blir utstyrt med en Bluetooth node på hver side av bilen. Renovatørene benytter IDBlue enheter til å lese brikkene i skapene når de henter avfallet. Brikkens ID sendes via Bluetooth til bilen, og tømmingen blir registrert. Rekkevidden på Bluetooth er som kjent begrenset, men om en renovatør skulle komme utenfor rekkevidde en liten stund vil IDBlue prøve å sende en stund før den gir opp. Vi går derfor ut fra at det ikke vil bli problemer med dette.

Dersom Agder Renovasjon velger å gå over til plastdunker for restavfallet, kan disse merkes med RFID brikker, og registreringen av tømmingen av disse kan da gå automatisk, slik som for papiravfallet.

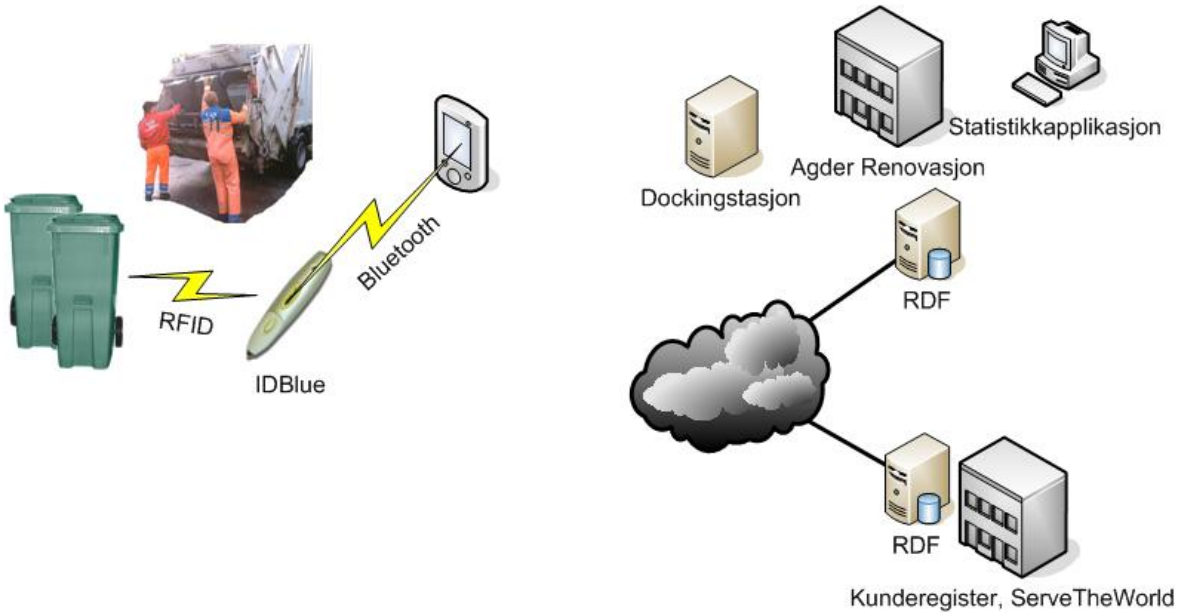
Vi foreslår også på sikt at alle bilene merkes med aktive RFID brikker. Alle bilene passerer vekten når de kjører inn eller ut av Heftingsdalen. Det kan derfor plasseres RFID lesere ved vekten som brukes til å registrere bilene når de kjører forbi. Dette kan være med på å koble mengden avfall til bil og kjørerute. For å bedre kontrollen på containerne bør disse også merkes med aktive RFID brikker. På denne måten kan det registreres hvilke containere som befinner seg i Heftingsdalen, og hvilke som er ute. I tillegg får en oversikt over hvilken bil som har kjørt hvilken container.

På samme måte som for det første forslaget foreslår vi at det også her benyttes en RDF database i bunnen. I motsetning til den PDA baserte implementeringen er det her ingen problemer med å benytte RDF i hele systemet. Vi vil derfor benytte RDF både på PCene i bilene og på serveren.

8 Implementasjon av demonstrator

8.1 Introduksjon

Vi har valgt å lage en enkel demonstrator med en RDF database i bunnen. Demonstratoren skal benytte en PDA og IDBlue til skanning av papirdunker, og den skal ha et program for uthenting av dataene.



Figur 9 - Oversiktsbilde, demonstrator

Som vi ser av figuren, vil informasjonen fra PDAen bli overført til RDF databasen ved hjelp av en dockingstasjon.

8.2 RDF

8.2.1 Database

I bunnen for demonstratoren ligger to MySQL servere, hvor den ene står på HiA mens den andre er hostet av ServeTheWorld [51]. På hver av disse serverene er en database. Disse databasene benytter vi, ved hjelp av Jena, som grunnlag for to RDF databaser. På HiA inneholder databasen all informasjon om rfid-brikker, sjåfører og tømminger, mens databasen hos ServeTheWorld inneholder en simulert kundedatabase. Dette har vi gjort for å kunne demonstrere hvordan to modeller på en enkel måte kan integreres.

Alt av SQL setninger blir automatisk generert av Jena, så vi trenger kun forholde oss til objekter med tilhørende metoder.

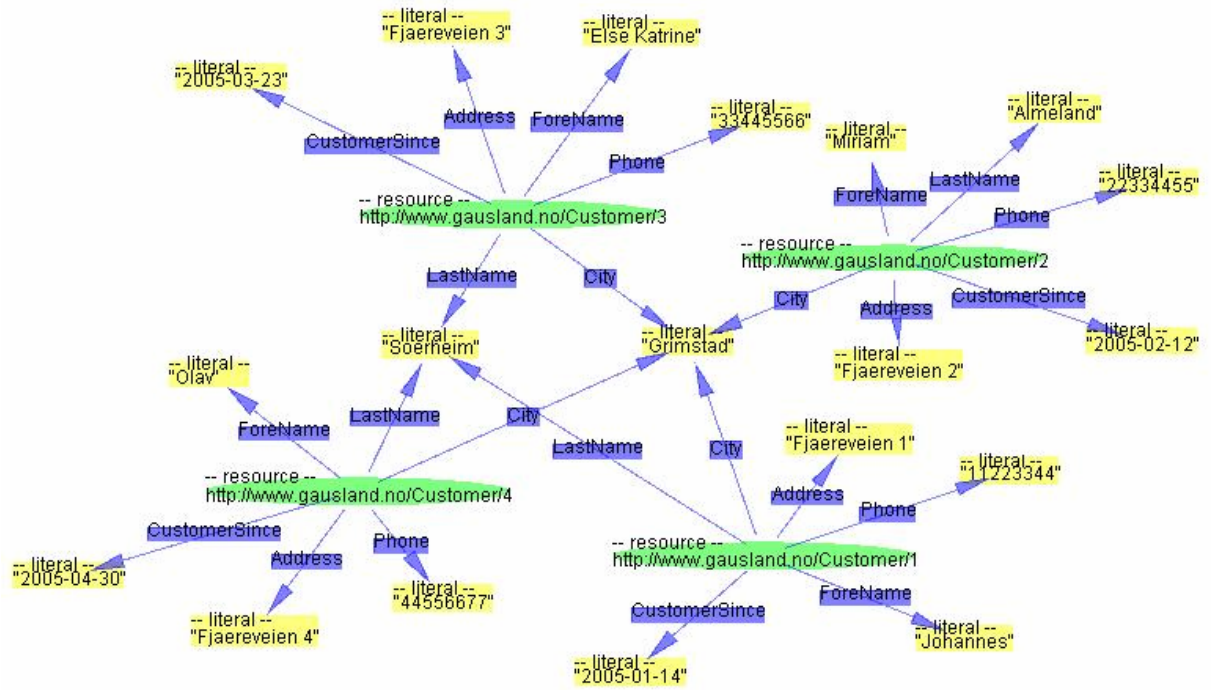
8.2.2 Vokabular

Kundedatabase

Kunder får en URI på formen <http://www.gausland.no/Customer/n>, der n er kundenummeret. Egenskaper som kan knyttes til kunde er blant annet ”ForeName”, ”Phone” og ”CustomerSince”. URI for egenskapen ”ForeName” er for eksempel <http://www.gausland.no/2005/03/customer#ForeName>. Et eksempel på en triple der

fornavnet på en kunde blir angitt blir da <http://www.gausland.no/Customer/1234> <http://www.gausland.no/2005/03/customer#ForeName> ”Kari”.

Eksempel fra vår applikasjon der vi har fire kunder med tilhørende egenskaper:



Figur 10 – Kundedatabase med fire kunder

AgderRenovasjon database

Her har vi benyttet <http://renovasjon.mine.nu/2005/02/waste#> som prefix for egenskaper og <http://renovasjon.mine.nu/AgderRenovasjon/> som prefix for ressurser. Egenskapene vi benytter er blant annet ”type”, ”Emptyed”, ”StopNumber”, ”Staff”, ”Name” og så videre.

Hver av RFID taggene har blant annet et stoppnavn, stoppnummer og kobling mot kunde ved hjelp av kundens URI. I tillegg har de en sekvens av tømminger kalt <rfid://xxx/Emptyed>, der xxx er den unike IDen til brikken. Hvert element i denne sekvensen er et såkalt ”tømmeobjekt” der URI inneholder tidspunkt for tømmingen. Tømmeobjektet inneholder informasjon som hvem som tømte og eventuell kommentar.

Hver av sjåførene har på samme måte en sekvens over hvilke ruter de har tømt. Disse rutene har igjen en sekvens over hvilke dunker som ble tømt. Dette er de samme såkalte tømmeobjektene som ble omtalt i forrige avsnitt. På denne måten blir det meste av informasjonen koblet på kryss og tvers. I et program kan vi da enkelt hente ut alle dunker som ble tømt en gitt dato og hvem som tømte dem. Eventuelt kan vi gå den andre veien og hente ut alle sjåfører, for så å hente fram hvilke dunker de har tømt.

Dunkene

Dunkene har to litteraler, en for stoppnavn og en for stoppnummer. De har også referanse til hvilken rute de tilhører og referanse til en sekvens med alle tømminger.

Tømming av dunken
Hver gang en dunk blir tømt, blir det opprettet en ressurs, her kalt dunktømming, med referanse til sjåfør. Dunktømmingen får også en litteral med dato, og eventuelt kommentar. Dunktømmingen blir lagt til i dunkens sekvens over alle tømminger og i en rutetømming sekvens over tømte dunker.

Ruten
Ruten har referanse til en sekvens med alle dunkene den inneholder og referanse til en sekvens med alle tømminger av ruten. I tillegg har ruten en litteral med navnet.

Tømming av ruten
Når en rute blir tømt, blir det opprettet en ressurs, kalt rutetømming, med informasjon om tømmingen. Rutetømmingen har en referanse til en sekvens med dunktømminger. Den har også referanse til hvilken sjåfør som tømte ruten.

Sjøføren
Sjåføren har en litteral med navnet og en sekvens med alle rutetømmingene vedkommende har utført.

8.2.3 Rutegenerering

For å generere filer med informasjon om hvilke dunker som skal tømmes, kjøres en RDQL spørring for å hente ut alle ruter fra databasen. For hver av rutene hentes det videre ut hvilke dunker som hører til, samt hvilken adresse og hvilket stoppnummer de har. Informasjonen blir skrevet til en tekstfil med et gitt format. Filnavnet angir hvilken dato den er generert samt hvilket rutenummer den inneholder. Ved hjelp av en fil med innstillinger, bestemmes intervallet eller tidspunktet programmet skal generere rutene på. Programmet genererer slik det er nå ruter med et intervall på 24 timer. På denne måten vil alle rutene genereres en gang for dagen. Det beste tidspunktet for generering av filer, vil være en liten stund før utkjøring. Når filene legges i mappen som synkroniseres ved hjelp av ActiveSync, vil de bli flyttet over på PDAen når den står i dockingstasjonen.

8.2.4 Rute- og dunkimportering

Når en rute er opprettet ved hjelp av PDA, genererer den to filer. Den ene filen inneholder informasjon om RFIDer med tilhørende avfallstype og adresse. Den andre filen inneholder informasjon om rutenummer, avfallstype og et navn på ruten. I tillegg inneholder den RFIDer med tilhørende stoppnummer. Når PDAen blir plassert i dockingstasjonen, blir disse filene overført til serveren.

På serveren står så to program som kjører med et gitt intervall, for eksempel en gang hver time. Når programmene oppdager filene med ruter og dunker, blir disse lest og gjort om til ressurser i RDF. Etter at informasjonen er importert, blir filene flyttet til et arkiv.

8.2.5 Importering av dunkregistreringer

Også etter at en tømmerunde er utført, blir det generert en fil. Denne filen inneholder informasjon om hvem som kjørte og en liste over alle RFIDene som skulle vært tømt.

```
http://renovasjon.mine.nu/AgderRenovasjon/Staff/2
E004010001112A26;2005-04-03T14:31:31.00+01:00;1;
E00401000111129E6;2005-04-03T14:31:29.00+01:00;0;Mangler tag
...
```

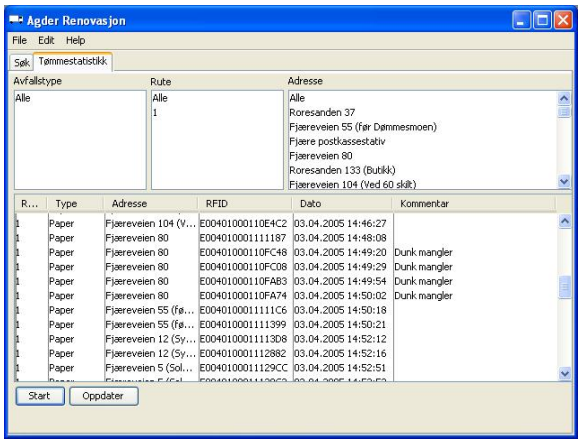
Figur 11 - Eksempel på tekstfil etter tømmerunde

Første linje inneholder URI til sjåføren som kjørte. Resterende linjer inneholder RFID, tidspunkt, 1 eller 0 og eventuelt en kommentar. Tidspunktet indikerer når brikken ble skannet eller en kommentar ble lagt til. Tallet indikerer om dunken ble tømt eller ikke, og kommentaren gir en forklaring på hvorfor tømming ikke ble utført. Tagger som ikke ble skannet og som ikke ble gitt en kommentar av sjåføren, vil få en standard kommentar ”Ikke tømt”.

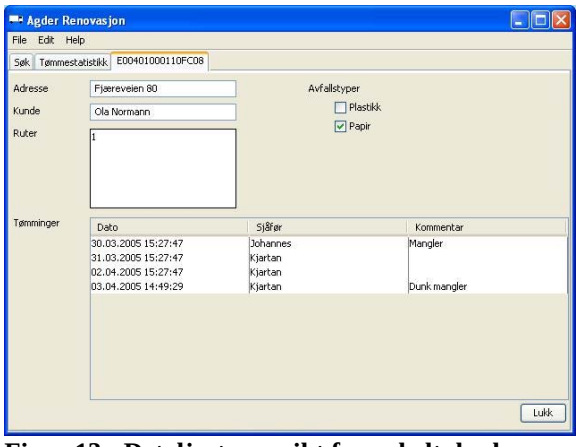
Når PDAen blir satt i dockingstasjonen, blir filen overført til serveren. På samme måte som importering av dunker og ruter, vil et program på serveren lese denne filen og importere informasjonen til RDF databasen.

8.2.6 Statistikk applikasjon

I tillegg til programmene som brukes til synkronisering av RDF database og PDA, har vi utviklet et lite program som henter ut enkle oversikter over utførte tømminger.



Figur 12 - Oversikt over siste tømming for alle dunker



Figur 13 - Detaljert oversikt for enkelt dunk

Ved klikk på ”Start” hentes all ruteinformasjon fra databasen. Brukeren kan velge avfallstype, rutenummer og adresse. I listen vises siste tømming for alle dunkene med eventuell kommentar. Ved å dobbeltklikke en spesifikk dunk, vil et nytt vindu (se Figur 13) med mer detaljert informasjon om alle tømminger for denne dunken dukke opp. For å illustrere kobling mellom to modeller har vi også lagt inn navnet på kunden.

8.3 RFID og PDA

8.3.1 Dunkregistrering

Programmet for registrering av tømming av dunker skal kjøres på en PDA, det skal gi brukeren nødvendig informasjon om ruten, og registrere dunkene som tømmes. Dersom brukeren glemmer en dunk skal det gis varsling, slik at feil kan rettes opp på et tidlig tidspunkt. All ruteinformasjon hentes fra databasen ved synkronisering, samtidig sendes informasjon om tømming tilbake til databasen.

I begynnelsen hadde vi planer om å benytte RDF på alle nivå i demonstratoren, men etter hvert viste det seg å bli ganske vanskelig. RDF rammeverket Drive [22], som også benytter seg av .Net, viste seg å være veldig vanskelig å benytte. Drive støttet ikke RQL eller RDQL spørringer, og rammeverket kunne kun benytte seg av filer på RDF/XML eller N-Triples format. Siden Drive hadde så få muligheter, valgte vi å la det være å bruke RDF på PDAen. I stedet valgte vi å benytte rene tekstfiler for informasjonen som skulle utveksles mellom PDA og databasen.

Som tidligere nevnt oppgraderte vi en iPAQ slik at den kjører Pocket PC 2003. Denne oppgraderingen skjedde sent i utviklingsperioden, og vi hadde derfor få muligheter til å teste på en PDA under utviklingen. På grunn av dette valgte vi å benytte en PC med Bluetooth til mye av testingen under utviklingen. Rammeverket .Net er ikke likt på PC og Pocket PC, slik at noen av funksjonene kun kan brukes på den ene plattformen. For å slippe å skrive dobbel kode opprettet vi en egen klasse med de funksjonene som er like for begge utgavene av rammeverket.

Grensesnittet på programmet er enkelt, slik at brukeren skal måtte gjøre minst mulig innstillinger for å benytte det. Ved oppstart prøver applikasjonen selv å koble seg opp mot den IDBlue enheten det sist var koblet til. I tillegg husker programmet hvem som var sjåfør sist det ble startet. Dersom hver bruker har sin PDA og IDBlue enhet vil det da kun være nødvendig å velge hvilken rute som skal kjøres. Når en dunk blir registrert gis det et lydsignal, slik at renovatøren skal få dette med seg uten å måtte se på skjermen. Lydsignalet som gis er forskjellig alt etter om alt er greit, eller om det mangler en dunk. Ved feil vises det også en melding på skjermen.

Demonstrator programmet ble etter hvert mer fokusert mot Pocket PC, siden den endelige versjonen skal benyttes på en PDA. På grunn av dette ble noen plattformavhengige funksjoner kun implementert i denne versjonen.

Når det oppdages mangler, blir det i tillegg vist et varslingsvindu på skjermen. Dette vinduet har en feilmelding, og ulike alternativer som renovatøren kan velge for å angi hvorfor dunken ikke ble tømt. Dersom renovatøren da ikke angir en feilmelding i løpet av en viss tid, blir det valgt en standard feilmelding.

Når tømmingen er ferdig for dagen, setter renovatøren PDAen sin i dockingstasjonen, og informasjonen blir automatisk lastet over til RDF databasen.

8.3.2 Ruteoppretting

Vi laget et eget program for oppretting av nye ruter. Brukeren angir rutenavn, rutenummer og hva slags rute det er. Det kan for eksempel være en rute med papirdunker, eller en rute med restavfall.

Programmet kobler seg automatisk til IDBlue enheten, og registrerer taggene som leses. Brukeren leser alle taggene på et stopp, og angir navnet på stoppet. Til slutt blir det laget filer med den informasjonen om ruten som er nødvendig.

På grunn av at brukeren må skrive inn litt tekst i forbindelse med registreringen, har vi valgt å lage dette programmet for PC og ikke Pocket PC, slik at det skal være lett å skrive inn navnene på stoppene.

8.3.3 Synkronisering

For å kunne utveksle informasjon mellom RDF databasen og de ulike programmene har vi laget et program som kjører på serveren. Dette programmet sjekker med jevne mellomrom om det har kommet noen filer med ny ruteinformasjon. I så fall legges informasjon om den nye ruten inn i RDF databasen.

På et gitt tidspunkt hver dag genererer applikasjonen filer med ruteinformasjon. Disse filene blir lagt i en mappe som synkroniseres med ActiveSync til en PDA. På den måten blir informasjonen automatisk lagt inn på terminalen når brukeren setter den i dockingstasjonen.

Filene fra terminalene, med informasjon om tømmingene, blir samtidig kopiert over til serveren. Disse filene blir lest, og innholdet blir gjort om til RDF og lagt inn i databasen.

8.3.4 Plassering av utstyr

Plassering av RFID utstyr:
Som tidligere nevnt har vi valgt å benytte bærbart RFID utstyr i forbindelse med utvikling av en demonstrator. Renovatøren vil derfor ha leseren med seg når han tømmer dunkene. Taggene må da plasseres slik at de kan avleses enklest mulig, men samtidig være best mulig beskyttet mot regn og fysiske påkjenninger. Slik som vi ser det har vi to alternativer for plassering av taggene. Enten under lokket på papirdunken, eller under kanten som går rundt dunken rett under lokket.

Vi utførte noen tester for å kontrollere rekkevidden når IDBlue skulle lese RFID brikker. Det ble testet rekkevidde gjennom tre, plast og metall. I tillegg testet vi hvordan lesingen fungerte når brikkene var montert på metall- og plast underlag.

På møtet med Agder Renovasjon den 23. februar, ble det testet at IDBlue klarte å lese en tag gjennom lokket på en dunk for papiravfall. For å lese gjennom lokket måtte vi holde leseren rett over der taggen er montert. For at dette ikke skal bli problem for renovatørene, må alle taggene monteres på nøyaktig samme sted, slik at de vet hvor de skal skanne dunkene. Alternativt kan taggene avleses fra innsiden av lokket.

Vinterstid dukker det opp ett nytt mulig problem. Når det snør, legger det seg snø oppå lokket. Dette fører til at snøen hindrer renovatøren i å skanne dunken fra utsiden. Ved å

montere taggene under kanten, vil ikke snø bli noe problem, men beskyttelsen mot fuktighet og dårlig vær blir ikke like god.

Vår anbefaling vil være at taggene monteres under lokket, og at renovatøren leser disse når dunkene henger på bilen med lokket åpent. Denne plasseringen gjør at avlesningen blir enkel, fordi renovatøren kan se taggene når han tømmer, samtidig som taggene vil være godt beskyttet når lokket er lukket. I februar monterte vi brikker på en dunk for å se hvor godt de hang fast. På denne dunken ble det montert brikke både under lokket, og under kanten på utsiden. Stedene hvor brikkene ble montert ble vasket og tørket før monteringen, slik at klisteret skulle få best mulig feste. Mot slutten av prosjektet, hang brikken på innsiden av lokket like bra, mens den under kanten hadde begynt å løsne litt.

Vi vil ikke anbefale Agder Renovasjon å ta i bruk taggene som ble brukt i forbindelse med testingen, dersom de velger å ta i bruk en RFID løsning. I så fall vil vi anbefale å bruke spesielle tagger som kan bli skrudd skikkelig fast på dunkene, slik at de ikke kan fjernes. Tre uker etter at vi monterte tagger på 65 dunker for å lage en testrute, hang fremdeles alle taggene fint, unntatt på seks dunker som stod på samme sted. Vi ser på det som usannsynlig at alle disse har falt av, siden alle andre hang godt fast. Vi antar at noen av de som bor i området rundt dunkene har fjernet disse.

Plassering av maskinvare:
I forbindelse med testingen valgte vi å ha alt utstyr plassert på høgskolen. Dette gjorde arbeidet med videreutviklingen lettere enn om utstyret hadde blitt plassert hos Agder Renovasjon. Renovatøren fikk med seg iPAQ og IDBlue enten samme morgen som han skulle tømme testruten, eller dagen før. Når tømmingen var ferdig, synkroniserte vi iPAQen med serveren, og kunne straks studere dataene.

For at et skikkelig system skal bli best mulig vil vi anbefale at hovedserveren plasseres hos en bedrift som leier ut servere, og inngå en service avtale som inkluderer at det tas regelmessige sikkerhetskopier. Plassering av hovedserveren hos en ekstern bedrift kan også bidra til at integriteten til dataene sikres, dersom Agder Renovasjon ikke gis full tilgang til dataene. Skal data gjøres tilgjengelig for andre vil det også være en fordel at serveren plasseres slik at den har raske linjer ut på internett, dette vil sannsynligvis være enklere å få til med en sentral plassering, fremfor plassering ute i Heftingsdalen hos Agder Renovasjon.

8.4 Forbedringsmuligheter

8.4.1 RDF

Slik programmet for generering fungerer nå, blir alle rutene generert hver gang programmet kjører. I en demonstrator som denne er ikke dette noe problem, men i en ferdig versjon bør kun de rutene som skal kjøres den aktuelle dagen bli generert. For å få dette til, må kalenderinformasjon knyttes til rutene. For eksempel kan det legges inn ukedag og tømmefrekvens eller alle datoene ruten skal tømmes.

Det kan også med fordel knyttes mer informasjon til RFID taggene. For eksempel kunne GPS posisjoner blitt lagt inn, for å kunne plotte ruten inn på elektroniske kart.

Når det gjelder programmet for uthenting av statistikker og oversikter, er det mange forbedringsmuligheter. Den informasjonen som er tilgjengelig nå, er et minimum for å

kunne demonstrere uthenting av data og sammenkobling av modeller. Av forbedringsmuligheter kan vi nevne kurver og diagrammer for visualisering, kobling av klager til kunder for bedre kundestatistikk og sammenhenger mellom klager og faktiske feil. Det bør også være enkelt å hente ut informasjon om hvilke dunker som eventuelt mangler, er ødelagt eller mangler RFID brikke.

Det bør også utvikles et program for oppretting og behandling av kunder, personell, dunker og ruter. For eksempel kunne et slikt program blitt brukt til å hente informasjon om hvilke kunder som ikke har betalt for så å automatisk ekskludere dunker fra rutene. Det kunne også under forutsetning av vekt på bilene, blitt lagt inn ruteoptimalisering basert på hvor det tømmes mest og minst.

Et alternativ til å lage alle delene av systemet vil være å integrere eksisterende løsninger mot dette. For alle programmene vil det også være en fordel å kunne utveksle all informasjon som RDF. Dersom vi hadde hatt et rammeverk for RDF på PDA, kunne dette blitt gjort ved å generere RDF/XML filer og synkronisere disse.

8.4.2 PDA

I en endelig versjon bør applikasjonen håndtere ”Power Notifications”. Nå har programmet problemer med å koble seg til IDBlue enheten igjen dersom PDAen blir slått av og på når programmet kjører. Vi brukte Platform Invoke (P/Invoke) slik at vi kunne fange opp meldinger om at enheten ble skrudd på eller av. Vi hadde problemer med å få dette til å fungere likevel, dette skyldtes både problemer med Bluetooth på PDAen, og vanskelig implementering av asynkrone hendelser i .Net Compact Framework. Vi bestemte oss derfor for å utsette denne finessen til en eventuell senere versjon.

Ved utvikling av applikasjoner for håndholdte terminaler må en fokusere på at batterikapasiteten er begrenset, så en må bruke minst mulig strøm. Et mulig alternativ her er å slå av skjermen når den ikke brukes. Ved å benytte P/Invoke kan en slå på skjermen hver gang brukeren trykker en knapp, og hver gang det skal gis en melding til brukeren.

8.5 Oppsummering

Vi har laget en demonstrator som benytter RDF på server siden. På grunn av mangelen på bra RDF verktøy til PDA, har vi ikke benyttet RDF her.

Vi har laget et eget RDF vokabular og program for å generere, importere og oppdatere ruteinformasjon. Filene som blir generert synkroniseres automatisk med PDA når denne settes i dockingstasjonen.

Programmet vi har laget for PDA benytter en RFID leser til å registrere dunkene etter hvert som de tømmes. Programmet viser oversikt over hele ruten, slik at det er lett å se hvor neste dunk er. Dersom en dunk glemmes varsles renovatøren med et lydsignal. Brukeren kan legge inn feilmeldinger på dunker som ikke er tømt, for eksempel at en dunk mangler RFID brikken. Ved neste synkronisering flyttes dataene om tømmingen til serveren, og programmet der konverterer dette til RDF og legger dataene inn i RDF databasen.

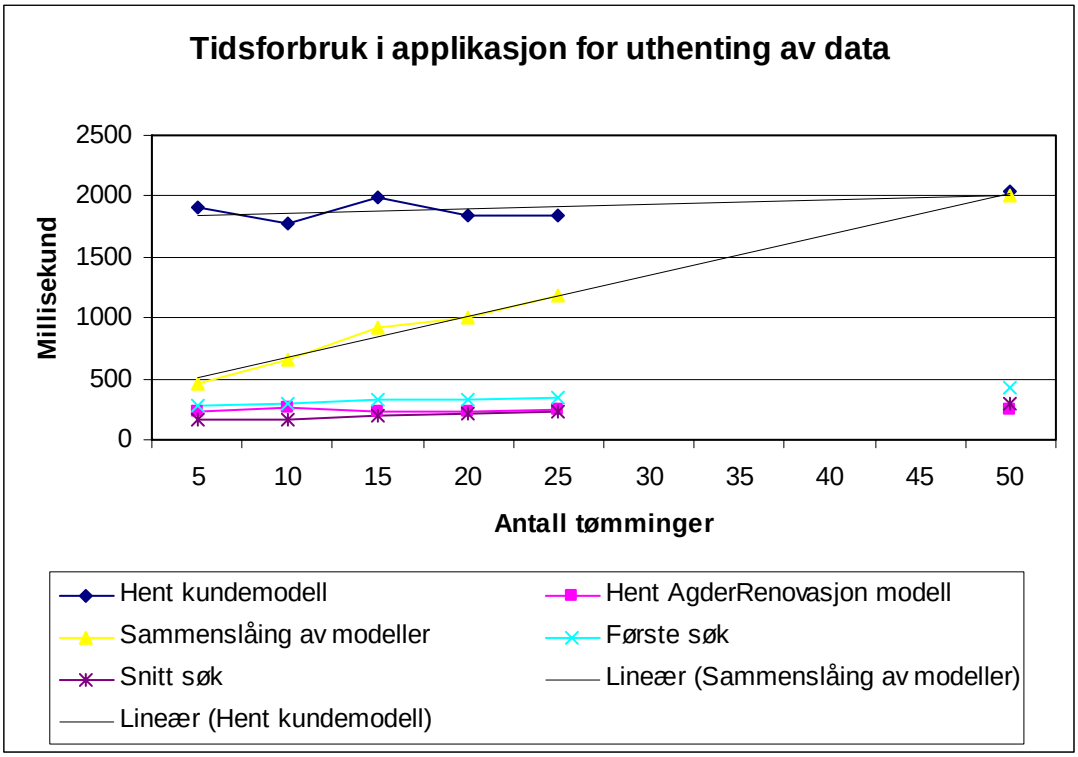
På serversiden, har vi laget et program som henter ut igjen denne informasjonen. Informasjonen kan begrenses ved å velge avfallstype, rute og adresse. Ved å gå inn på en enkelt dunk, listes all informasjon om denne, i tillegg til sjåfører og kommentarer.

9 Resultat

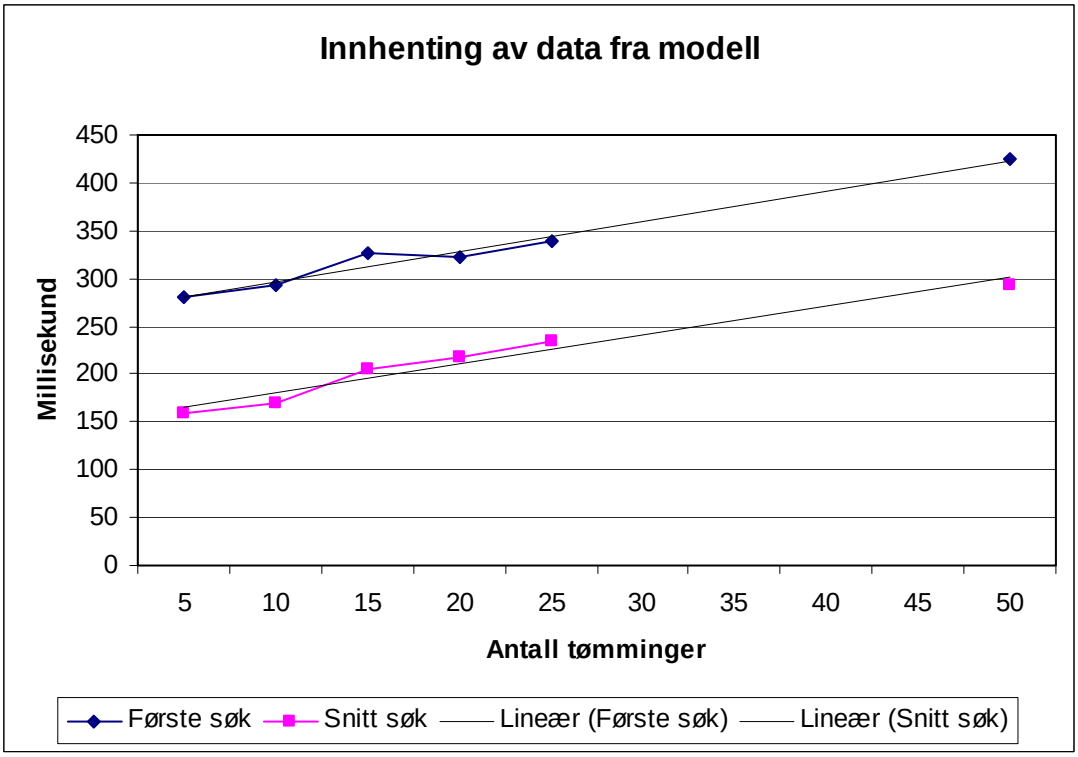
9.1 RDF

Vi har laget en samling av programmer som fungerer sammen. Vi har et program som importerer RFID informasjon, et program som importerer ruteinformasjon, et program som genererer filer til PDA og et program som legger informasjon om tømminger fra PDA inn i databasen. Serveren produserer filer som synkroniseres med PDA, mens PDAen produserer filer som synkroniseres tilbake til server. Vi har også utviklet et enkelt oversiktsprogram som henter ut igjen denne informasjonen. Til alle IDene kan det knyttes en kunde URI. Ved å slå sammen modellen i AgderRenovasjon med kundedatabasen, vil vi i tillegg få inn navn, adresser og annen kunderelatert informasjon. Eventuelle endringer må gjøres på de opprinnelige modellene.

Vi har gjort noen målinger på tidsforbruk på uthenting av data fra modellene. Det vi har målt er tiden det for å koble seg opp til databasene, hvor lang tid det tar å sette sammen modellene og hvor lang tid det tar å hente ut all relevant informasjon. Målingene er utført etter 5, 10, 15, 20, 25 og 50 tømminger på testruten med 65 dunker.



Figur 14 – Graf som viser tidsforbruk ved oppkobling, sammenslåing og uthenting av data



Figur 15 – Graf som viser tidsforbruk ved uthenting av data

Å koble seg opp mot modellene tok omtrent like lang tid hver gang, uavhengig av datamengde. Det var likevel en liten tendens til økning. For den lokale databasen lå tidsforbruket for oppkobling på ca 200 ms, mens det tok ca 2000 ms å koble seg opp mot kundedatabasen.

Også uthenting av data ble lite påvirket av datamengden. Som vi ser av det siste diagrammet, er det en liten økning i tidsforbruk som øker jevnt med den økte datamengden.

Den største økningen i tidsforbruk i forhold til datamengden, var sammenslåingen av modellene. Som vi ser av det første diagrammet, stiger denne kurven ganske jevnt og bratt sammenlignet med de andre kurvene.

Etter 10 tømminger av testruten er den største tabellen i databasen på ca 4000 rader, mens det etter 50 tømminger er ca 17000 rader. For å teste ytelsen ved større mengder data, genererte vi flere tømminger slik at databasen nærmet seg 250000 rader. Dette resulterte i ”memory out of range exception” ved sammenslåing av modellene. Vi kuttet ut sammenslåingen for å se om vi da kunne hente ut informasjon. Nå ble det ikke problemer med minnet, men uthentingen tok lang tid. Etter å ha sett nærmere på koden, viste det seg at problemet var metoden for sortering. Ved å kommentere bort denne metoden og la resultatet komme usortert, fikk vi presentert data fra modellene innen rimelig tid.

Vi har ikke utført ytelsestester for de ulike alternativene til RDF programvare. Derimot viser vi til resultatene i [52] som er en skalerbarhetsrapport skrevet av Ryan Lee, 26 juli 2004. Her er blant annet responstidene for ulike ”triple store” sammenlignet.

9.2 RFID

Tilkoblingen mellom IDBlue enheten og iPAQ tok stort sett 3-10 sekund, men noen ganger tok det 60 sekund før koblingen ble opprettet. De gangene oppkoblingen tok 60 sekund virket det som om driveren til IDBlue låste seg dette minuttet. Vi opplevde aldri oppkoblingstider mellom ca 15 sekund og 59 sekund. Etter litt undersøkelser ble det slått fast at det er metoden AutoConnect i driveren til IDBlue som av og til bruker 60 sekunder. Denne metoden er den som sørger for at programmet kobler seg sammen med IDBlue. Grunnen til at denne metoden av og til brukte mye lenger tid enn ellers klarte vi imidlertid ikke slå fast.

IDBlue skrur seg vanligvis av etter 2 minutter uten bruk. iPAQen kobler seg automatisk opp mot enheten igjen når denne blir slått på igjen. Så lenge IDBlue ikke var slått av i mer enn 2-3 minutter fungerte dette, men dersom det gikk lenger tid koblet ikke enhetene seg sammen igjen. Eneste muligheten til å gjenopprette koblingen når dette skjedde, var å starte programmet på nytt. Vi prøvde også å fjerne driveren og opprette den på ny dersom koblingen var nede i mer enn 2 minutt, men dette fungerte dessverre ikke.

For å redusere problemet økte vi tiden det skulle ta før IDBlue skrudde seg av. Den største tiden vi kunne velge var 4 minutt og 15 sekund. Med denne tiden hadde vi ikke problemer når vi testet ruten selv. Når to renovatører fra Agder Renovasjon testet alene ble det problemer med oppkoblingen etter en stund. Dette skyldtes enten batterikapasiteten eller at tidsintervallet for å slå av ble for kort, slik at enheten slo seg av for lenge.

Programmet for å opprette nye ruter ble laget for PC, og ikke PDA. Med PC versjonen av driveren til IDBlue hadde vi ingen problemer med å koble opp igjen etter at IDBlue hadde vært frakoblet. Oppkoblingen gikk stort sett også raskere enn på en PDA, men det kunne av og til gå opptil 10 sekunder å koble til IDBlue etter at den hadde vært avslått en stund.

Etter å ha vært i kontakt med Cathexis Innovations Inc. testet vi oppkoblingen mellom en ny Qtek 9090 og IDBlue. Oppkoblingen fungerte da fint selv om IDBlue var avslått i mer enn 5 minutter. Tiden det tok å koble opp var like lang som for iPAQ og IDBlue.

Batterivarigheten til IDBlue skal i følge Cathexis Innovations Inc. være 4 til 6 timer ved vanlig bruk. Under utviklingen skannet vi veldig mange RFID brikker i løpet av en dag uten at det ble problemer med batterikapasiteten. Denne skanningen var stort sett slik at vi skannet mange brikker på relativt kort tid, og IDBlue var avslått ellers. Når vi testet ruten sammen med en renovatør, opplevde vi at IDBlue signaliserte at det var lavt batterinivå etter litt over en time. Dette var imidlertid etter at vi hadde justert opp tiden det skulle ta før enheten slo seg av, slik at den hadde stått på konstant i litt over en time. Senere testet to renovatører uten at vi var med, og da fikk vi beskjed underveis at det var lite batteri igjen. Når renovatørene kom tilbake med utstyret var batteriet på IDBlue helt tomt. Batterivarigheten på PDAen var imidlertid ikke noe problem.

Rekkevidden på IDBlue er ca 5 cm, slik leverandøren har oppgitt. Rekkevidden ble noe redusert i tilfeller der det befant seg noe mellom RFID brikken og IDBlue. Leseren hadde ikke problemer med å lese brikker gjennom en treplate på ca 2,5 cm, men for å lese gjennom dette måtte leseren og brikken være helt inntil platen. Det var heller ikke problem å lese RFID brikker gjennom lokket på plastdunkene til papiravfallet. Plasserte vi metall mellom RFID brikken og IDBlue ble det umulig å lese brikken, det samme gjelder også når

brikken ble plassert på et objekt av metall. Samme hvor kort avstanden var så klarte vi ikke lese brikkene i disse tilfellene.

I forbindelse med testingen brukte vi ikke spesialtilpassede RFID brikker, men helt standard brikker. Disse brikkene fungerte bra sammen med IDBlue. Brikkene var laget som klistremerker slik at de kunne festes enkelt. Disse brikkene overlevde perioder med 5-10 kuldegrader uten å ta skade. Klisteret hadde også brukbar festeevne. Etter 3 uker hang fremdeles brikkene godt på dunkene. Eneste unntaket var brikkene på seks dunker ved samme stopp, her manglet alle brikkene.

Renovatøren fra Agder Renovasjon som testet utstyret, syntes det var tungvint å gå rundt med dette. Han mente det hadde vært mye greiere med utstyr som var fast montert på bilen slik at han slapp å gå og bære på utstyret.

9.3 PDA

Resultatet på PDA siden er et program for bruk under tømming av papirdunker. Programmet er laget slik at brukeren kun trenger å gjøre noen få innstillinger. De eneste innstillingene brukeren trenger å gjøre, er å velge sjåfør og hvilken rute som skal kjøres. Programmet merker av at dunkene er tømt etter hvert som de blir registrert. Listen over dunkene på ruten ruller automatisk slik at siste tømte dunk og de neste på ruten alltid vises på skjermen.

Når en dunk blir registrert, gis det et lydsignal som informerer renovatøren om at dunken er registrert. Det sjekkes også om det mangler registrering på dunker tidligere på ruten. Dersom det er noen som ikke er registrert, varsles renovatøren om dette med et annet lydsignal. Samtidig kommer det i så fall opp navnet på dunken på skjermen, og renovatøren får mulighet til å legge inn en feilmelding på hvorfor dunken ikke er tømt. Selv med maksimalt volum på PDAen, er lydstyrken på signalene så lavt at det lett kan bli overhørt i støy fra motoren eller annen trafikk.

Under testene var Bluetooth aktiv hele tiden på grunn av koblingen mot IDBlue. Vi hadde likevel ingen problemer med batterikapasiteten på PDAen. Batteriet gav imidlertid ikke nok strøm til å kunne bruke PDAen en hel dag. Etter ca 3 timer var det 30% igjen av batterikapasiteten.

10 Drøfting

10.1 RDF

10.1.1 Vår implementasjon

Løsningen vi først valgte med et program for hver oppgave fungerte greit. Ulempen da var at programmene hver for seg hadde en kobling til modellene. Siden har vi laget et slags skall rundt disse programmene som oppretter en kobling til modellen, for så å starte de andre programmene i egne tråder. På denne måten opprettes koblingen kun èn gang. Dette resulterte i et nytt problem ved at alle programmene prøvde å oppdatere modellen samtidig, noe som igjen førte til deadlock. Dette ble unngått ved å bruke en synkronisert variabel rundt de partiene som jobbet mot modellen, noe som fungerte veldig godt. I ettertid har vi sett at Jena har innebygget støtte for å sikre eksklusiv tilgang til modeller. Ved å benytte metodene ”enterCriticalSection” og ”leaveCriticalSection” med parameterene ”ModelLock.READ” eller ”ModelLock.WRITE”, ville problemene med samtidighet ikke oppstått.

All informasjon til og fra PDA går nå som ren tekst. Dette fører til at vi på serversiden må generere filer på dette formatet, og lage funksjoner for å importere tilbake til RDF. Det hadde vært en stor fordel å latt all informasjonen gå som RDF, også mellom PDA og Server. Årsaken til at vi likevel har valgt å benytte ren tekst, er den manglende støtten for RDF på PDAen. En mulighet hadde vært å likevel benytte RDF for så å skrive kode for å lese dette formatet selv. Vi måtte da også ha skrevet kode for å generere filer på RDF formatet. Dette ville gjort arbeidet på serveren noe enklere, i og med at Jena har innebygd støtte for å importere informasjon fra RDF/XML og kan samtidig enkelt generere RDF/XML filer. Det ville ført til mye mer arbeid på programmet for PDAen.

Som vi så i diagrammet for tidsforbruk, ble ikke oppkoblingen påvirket av datamengden i særlig grad. Dette tyder på at det kun blir opprettet en kobling mot modellen, uten at modellen blir sendt i retur. Etter å ha analysert kommunikasjonen mellom programmet og MySQL serveren, viste det seg at dette stemmer. Så snart to modeller slås sammen ved hjelp av ”union”, blir all informasjon hentet ut fra databasene og lagret lokalt i minnet. Dette kan ta svært lang tid og føre til stort minneforbruk. Dette ser vi også av diagrammet som viser at tidsforbruket for sammenslåingen øker kraftig for hver tømming av ruten. Resultatet etter testen med 250000 rader, som førte til ”memory out of range exception”, bekrefter stort minneforbruk. Dersom modellen ikke hentes inn i programmet på forhånd, vil Jena kjøre SQL spørringer etterhvert som ressursene hentes ut. Dette kan resultere i mange spørringer mot databasen. Union fungerer derfor best på små modeller. På store modeller bør informasjonen hentes ut etterhvert som den trengs. En ulempe med programmet vi har laget for uthenting av data, er at vi henter ut all informasjon om alle ressursene, og gjør disse om til objekter. Store modeller vil derfor ta svært lang tid å hente inn, og samtidig vil det føre til stort minneforbruk. I en fremtidig utgave bør kun den informasjonen som er interessant til en hver tid hentes ut.

10.1.2 Jena

Det kan diskuteres om Jena var et godt RDF verktøy i vårt tilfelle. Vi har ikke hatt nevneverdige problemer med Jena, men det finnes noen ulemper. Et eksempel er mangelen på brukerstyrt tilgang til modellene. Joseki, som er webgrensesnitt for Jena, har heller

ingen mulighet for styring av tilgang på brukernivå. På denne måten blir modellene tilgjengelige for alle. Ofte er en del av meningen med RDF å gjøre informasjon tilgjengelig for alle på en strukturert måte (Semantic Web). Det hadde likevel i vårt tilfelle vært greit å kunne begrense tilgangen på tilsvarende måte som i Sesame, der APIen tilbyr innlogging med brukernavn og passord.

Når det gjelder lagringsformatet, har Sesame som nevnt tidligere valgt en noe annen løsning enn Jena. I Sesame er det en egen tabell med indekser og URIer. Indeksene blir brukt i resten av tabellene istedet for de lange URIene. Dette reduserer størrelsen på databasen betraktelig, men fører til flere koblinger mellom tabellene ved uthenting av data.

En fordel med Jena derimot er at det ikke trengs noe ekstra programvare installert på serveren enn databasen som skal benyttes i bunn. Med Sesame kreves TomCat eller en annen servlet kontainer installert på serveren. Programvaren vi har utviklet kan på en enkel måte distribueres som en kjørbar .jar fil med Jena inkludert. Tilsvarende løsning kan benyttes i forbindelse med Sesame. Selv om det i etterkant har vist seg at Sesame har noen fordeler framfor Jena, har erfaringene våre vært at Jena er et godt RDF verktøy.

Noe som var litt forvirrende, var de mange ”doble” metodene. For å hente en ressurs, finnes det en metode ”getResource”. Dersom ressursen som etterspørres ikke finnes, blir ressursen opprettet. Samtidig finnes det en metode ”createResource” som oppretter ressurser. Dersom en prøver å opprette en ressurs som finnes fra før av, vil istedet denne bli returnert. På denne måten kan de to metodene brukes omtrent om hverandre.

Selv om vi i denne oppgaven ikke har fått noen særlig nytte av Joseki som webgrensesnitt mot Jena, ser vi fordelene et slikt grensesnitt kan gi. For eksempel kan applikasjoner som skal kommunisere med Jena utvikles i andre språk enn Java ved å sende forespørsler mot et webgrensesnitt.

10.1.3 Modenhet og muligheter

Siden RDF først ble lansert i oktober 1997, har det blitt utviklet en rekke løsninger for bruk av RDF. De fleste av disse er basert på åpen kildekode men mange er fremdeles i tidlige og ufullstendige versjoner. Noen av løsningene kan likevel tas i bruk. De mest brukte er da Jena med spørringsspråket RDQL og Sesame med spørringsspråket SeRQL.

Det finnes i dag lite kommersiell programvare som tar i bruk RDF. Vi ser likevel noen som benytter rammeverket, blant annet Adobe som bruker RDF i sitt XMP (Extensible Metadata Platform) format. Det finnes også andre program som benytter RDF, men de er ofte utviklet av de samme som utvikler løsningene for RDF. Ofte er disse også del av forskningsprosjekt. Et eksempel på dette er Open Directory [53].

En fordel med RDF, er at ressursene uten problem kan ha varierende antall egenskaper. I en standard SQL database må alle radene inneholde den samme informasjonen. Alternativt vil noen kolonner stå tomme, og skape problemer ved uthenting. RDF har også IDer som har mening utenfor databasen. I en standard SQL database vil for eksempel IDen 363 ikke ha noen mening andre plasser enn lokalt i databasen. Når vi skriver spørringer for RDF, vil disse bli oversatt til SQL spørringer. Å skrive spørringene direkte i SQL selv, vil derfor sannsynligvis gi raskere respons. Vi har også merket at det tar tid å både legge inn og hente ut data i Jena. I en tidskritisk applikasjon kan det derfor bli problem å bruke RDF. En fordel med SQL er også at dette er et komplett språk for spørringer mot databaser, og er

standarisert. I RDF finnes det mange språk der alle er gode på sin måte, men ingen kan kalles komplette.

Den største fordelen med RDF ligger i muligheten for Enterprise Information Integration. Ved å la ulike modeller ligge på ulike servere, gjort tilgjengelig for de som skal ha tilgang, kan brukere selv hente og koble den informasjonen de ønsker. I en tenkt situasjon for renovasjonsbransjen, kunne for eksempel alle kundedata vært utilgjengelig for andre, mens statistikker på avfall og klager kunne vært tilgjengelig for miljøverndepartement og liknende. Dersom alle renovasjonsselskap bruker de samme modellene, vil det være enkelt å integrere disse dataene for å hente ut samlet statistikk. Dette vil i så fall kreve at RDF systemet har korte responstider og kan behandle store mengder data.

10.2 RFID

Tilkoblingstidene mellom IDBlue og iPAQ var stort sett tilfredsstillende. 3-10 sekunder oppkoblingstid er normalt i forbindelse med oppretting av en Bluetooth kobling. Tiden det tok å koble opp ville blitt mye større dersom vi først skulle søkt etter Bluetooth enheter, og deretter valgt en enhet. Men siden vi koblet opp direkte mot MAC adressen til IDBlue enheten gikk oppkoblingen raskt. Unntaket var de gangene oppkoblingen plutselig tok 60 sekund. Grunnen til at driveren til IDBlue av og til brukte så lang tid er ukjent. Vi fant aldri noen sammenheng til noe som kunne ha innvirkning på dette. De gangene oppkoblingen tar lang tid er det kun metoden AutoConnect i driveren som bruker lang tid, resten av koden utføres på normal tid.

Det meste med IDBlue fungerte. Vi fikk koblet oss til den, lest RFID brikker og fikk koblet til igjen etter at den hadde vært avslått. Det eneste problemet var at den ikke måtte ha vært slått av for lenge. Siden det meste fungerte bra mistenkte vi at det var implementeringen av driveren til IDBlue som stod for de problemene som var. Vi prøvde derfor forskjellige forsøk på fjerne driveren og legge denne til på ny, men dette fungerte ikke. Etter en del forskjellige forsøk fant vi ut av vi ikke kunne bruke mer tid på å få dette til siden vi kun skulle utvikle en demonstrator. Oppjusteringen av hvor lang tid det skulle ta før IDBlue slo seg av reduserte litt av problemet siden IDBlue da ikke slo seg av mellom de ulike stoppene. Dette hjelper imidlertid lite når batterikapasiteten begynner å bli liten, da reduseres tiden slik at enheten slår seg av mye raskere.

Vi er ikke fornøyd med kapasiteten på batteriet som sitter i IDBlue. Vi kunne riktignok skanne mange dunker dersom dette ble gjort i korte perioder og enheten var avslått mellom disse periodene. Oppjusteringen av tiden for automatisk deaktivering førte til at enheten stod på i lengre perioder enn den ville gjort ellers, og brukte derfor opp batteriet raskere. Vi synes likevel at driftstiden burde vært godt over 2 timer. Testruten inneholder kun 65 dunker, så det er ikke skanningen av disse som bruker opp batterikapasiteten. Det som trekker det meste av strømmen må da være Bluetooth koblingen.

Vi tok etter hvert kontakt med Cathexis Innovations Inc. for å høre om de kunne forklare hva som forårsaket at oppkoblingen av til IDBlue ikke fungerte dersom enheten hadde vært avslått i mer enn 2-3 minutter, når alt annet fungerte greit. Etter litt kommunikasjon fant vi ut at det kunne være noe med iPAQen som forårsaket disse problemene. For å sjekke dette fikk vi lånt en ny Qtek 9090 som vi kjørte et testprogram på. Oppkoblingen mot IDBlue virket da fint, uavhengig av hvor lenge enheten hadde vært avslått. Testperioden var dessverre ferdig når vi oppdaget at problemene skyltes iPAQen, vi disponerte dessuten kun en PDA under testperioden slik at det ikke ville vært mulig å teste med en annen PDA.

Hadde vi hatt en annen PDA under testingen ville vi kunnet hatt et kortere intervall på hvor lenge IDBlue stod på før den slår seg av. Dette ville imidlertid fått frem et annet problem, renovatørene måtte enten slått på IDBlue litt før de kom til en ny dunk, eller så måtte de stått ved dunkene og ventet på at Bluetooth koblingen ble etablert på nytt. Hadde oppkoblingen vært raskere ville dette fungert godt, samtidig som vi kunne spart mye batterikapasitet.

Under testingen hadde vi ikke problemer med rekkevidden til IDBlue. Bærbart utstyr gjør at det blir ganske naturlig for brukeren å ta IDBlue pennen bort til RFID brikken. Avlesningen går så raskt at en kun trenger å ta pennen raskt borttil brikken. Det er derfor enkelt å lese av brikkene på dunkene på kort tid.

RFID brikkene vi benyttet fungerte overraskende bra. Disse brikkene var som tidligere nevnt helt standard brikker som var laget som klistremerker. På forhånd hadde vi fryktet at disse ikke kom til å henge på hele testperioden, men det viste seg at de hang bra på dunkene. Unntaket var seks dunker på samme sted, men vi ser på det som svært lite sannsynlig at alle disse skulle ha falt av når ingen andre på ruten hadde forsvunnet. Vi har derfor gått ut fra at det mest sannsynlig er noen som har fjernet disse brikkene. Samme hvor bra brikkene har fungert til testingen ville vi valgt spesial brikker til et mer permanent prosjekt. Det eksisterer blant annet brikker som kan limes fast på bra måter, og brikker som kan festes til dunkene med for eksempel popnagler. Slike brikker vil sitte mer permanent på dunkene, slik at en slipper problemer med dunker som mangler RFID brikker.

Vi valgte å benytte bærbart utstyr til testingen, og tror dette var et smart valg. Skulle vi brukt stasjonært utstyr i bilene måtte vi brukt mye tid på montering av utstyret. Utviklingen og testingen ville også blitt vanskeligere når utstyret sitter på en bil til Agder Renovasjon. Med bærbart utstyr var det ingen problemer med å bruke utstyret selv i testing og utvikling, og overlevere det til renovatørene når de skulle teste det. Vi var imidlertid enige med renovatørene i at det vil være tungvint å måtte bære rundt på utstyret.

Tiden det tar å lese brikkene på dunkene med IDBlue er som nevnt liten, en vil derfor ikke nødvendigvis spare noe særlig tid på tømmingen ved å benytte stasjonært utstyr. Stasjonært utstyr vil derimot gi andre fordeler, slik som bedre pålitelighet på avlesningene. En renovatør kan glemme å skanne brikken på en dunk, men stasjonært utstyr vil automatisk registrere alle dunkene som henges på bilen.

10.3 PDA

Vi vil påstå at selve programmet for bruk under tømming er brukervennlig, spesielt siden det kun skal velges sjåfør og kjørerute. Siden det automatisk rulles slik at siste tømte dunk er midt på skjermen vil det være relativt kjapt å se hva som er neste dunk på ruten dersom renovatøren er usikker på dette. Brukervennligheten blir imidlertid delvis ødelagt på grunn av problemene med koblingen til IDBlue. Hadde koblingen til IDBlue fungert bedre ville vi ha fått et mye mer brukervennlig resultat.

Brukervennligheten i programmet kan forbedres ytterligere ved at hver bruker har en egen PDA. I så fall slipper brukeren å velge navnet sitt i en liste, og ved synkronisering hentes automatisk kun den ruten som renovatøren skal kjøre. Brukeren trykker i så fall bare en knapp når han er ferdig med ruten.

Lydsignalene som gis renovatøren er litt for svake. Det ville vært ønskelig å ha et kraftigere signal, slik at renovatøren ikke kan gå glipp av et signal. Slik det er nå risikerer vi at en renovatør blir usikker på om en dunk er registrert eller ikke. Han kan da måtte bruke tid på å finne frem PDAen fra en lomme for å sjekke om dunken ble registrert. Alternativt kan han skanne dunken på ny, for å høre om dunken blir registrert da. Usikkerhet på om dunker er registrert eller ikke vil uansett føre til økt tidsbruk.

Andre PDAer kan ha kraftigere høyttaler enn den vi brukte i denne oppgaven, slik at styrken på lydsignalene ikke blir noe problem. Det kan alternativt benyttes øreproppper, slik at en er sikker på at renovatøren får med seg lydsignalene. For renovatører som kjører alene kan det også være interessant å benytte øreproppene til å spille musikk. Dette vil i så fall føre til økt strømforbruk, noe som kan føre til problemer med å ha nok batterikapasitet.

Batterikapasiteten opplevde vi ikke som noe problem under testene, men det at det kun var 30 % igjen etter 3 timer drift viser at batterikapasiteten med denne PDAen ikke vil være nok til en hel arbeidsdag. Hadde Bluetooth vært aktiv mindre av tiden ville vi kunne spare en del strøm.

I likhet med renovatøren som testet utstyret synes også vi at det blir tungvint å gå rundt med en PDA. En ny og mindre PDA ville sannsynligvis vært mye enklere å ha med seg. Det som kan være mer problem er at renovatøren må ta frem PDAen for å kunne se skjermen. Vi har ikke benyttet GPS i forbindelse med demonstratoren, hadde vi benyttet dette ville enheten blitt enda større. Bruk av GPS ville dessuten ført til at brukerne ville kommet til å sett mer på skjermen, blant annet for å sjekke hvor de befinner seg i forhold til neste dunk på ruten. Med utstyr montert i bilen vil det ikke være problematisk å se på skjermen, da denne kan monteres slik at den er lett tilgjengelig fra førerplassen.

10.4 Videre arbeid

I denne oppgaven har vi ikke utviklet et ferdig system, kun en demonstrator. Skal det utvides til et skikkelig system vil dette begynne med at Agder Renovasjon bestemmer seg for hva de vil ha. Neste steg blir å kjøpe inn nødvendig utstyr til å kunne utvikle kvalitetssikringssystemet. Det er på dette tidspunktet ikke nødvendig å kjøpe inn utstyr til alle bilene. Det som trengs i begynnelsen er utstyr til minst en bil, samt utstyret som trengs på server siden.

Under utviklingen er det viktig å få til et bra samarbeid med renovatørene for å få til en mest mulig brukervennlig løsning. Det er viktig at systemet er brukervennlig, slik at det ikke blir sett på som en ekstra belastning, men som en hjelp til å utføre arbeidet på en best mulig måte.

Når en tilfredsstillende løsning er klar og testet, kjøpes det resterende utstyret inn og monteres på de resterende bilene. Alle dunkene som skal være med i systemet må også merkes med RFID brikker, og GPS posisjonen registreres. Rutene som skal kjøres må opprettes, og dunkene på rutene registreres på rett rute.

10.5 Vår anbefaling

Tenkt scenario
Som tidligere nevnt er ikke dette et reelt forslag vi anbefaler Agder Renovasjon, men litt tanker om hvordan et kvalitetssikringssystem for avfallshåndtering kan fungere. Denne løsningen ville vært dyr å implementere i fullskala.

Forslag 1, PDA basert implementering
Denne løsningen gir renovatørene tidlig tilbakemelding dersom det blir gjort en feil. De får da muligheten til å rette opp feilen tidlig, slik at kostnaden med å rette feilene blir minst mulig. Ved å benytte denne løsningen sammen med GPS og elektroniske kart kan det gis geografisk informasjon til sjåførene. Dette kan være en stor fordel i opplæringen av nye sjåfører. Forslaget innebærer at renovatøren må gå med en PDA på seg, dette kan være ganske tungvint når en skal ut og inn i bilen hele tiden.

Forslag 2, PC basert implementering
Denne løsningen vil være mer omfattende enn det første forslaget, men arbeidet til renovatøren blir også enklere. Med PC montert i bilen kan sjåføren alltid ha kartet fremme på skjermen, slik at han ser kjøreruten, uten å måtte ta frem en PDA. RFID leser på bilen fører også til at registreringen av papirdunkene går automatisk, slik at renovatøren slipper å tenke på dette. For de typene avfall som ikke tømmes på samme måte som papiravfallet må renovatørene benytte bærbart utstyr til avlesningen av RFID brikkene, men siden IDBlue kommuniserer trådløst via Bluetooth er det ikke nødvendig at renovatørene har med seg mer utstyr enn akkurat denne enheten.

Anbefaling
Vi vil anbefale Agder Renovasjon å starte med en enkel variant av forslag 2. Vi vil anbefale at systemet først implementeres i en enkel versjon kun for papirdunker. Dette vil gi selskapet muligheten til å vurdere fordeler og ulemper med systemet før de eventuelt bestemmer seg for å ta med andre typer avfall i systemet. I tillegg vil utviklerne kunne få en del tilbakemeldinger på hva som fungerer bra, og hva som fungerer mindre bra, slik at dette kan forbedres før det tas i bruk på flere områder.

I første omgang vil vi kun benytte PCer i bilene som er tilkoblet GPS og fastmontert RFID leser. Vi tar ikke med GPRS til å begynne med, og reduserer på denne måten kostnadene, siden vi ikke synes det er kritisk å kunne sende oppdateringer underveis. Det vil være godt nok å synkronisere informasjonen når bilene er i Heftingsdalen. Siden vi foreslår å starte med kun papiravfall vil det heller ikke være nødvendig å montere Bluetooth noder på bilene.

Grunnen til at vi anbefaler å starte med nettopp papiravfall er at dette avfallet allerede samles inn i plastdunker. Måten disse dunkene tømmes på muliggjør automatisk registrering av dunkene, slik at det ikke blir noe ekstra arbeid for renovatørene. Dessuten er antall dunker for papiravfall mindre enn antall abonnenter, slik at kostnaden med å implementere kvalitetssikringssystemet på alle dunker av en type er mindre for papiravfall enn for andre typer avfall.

Grunnlaget for at vi anbefaler forslag 2 fremfor forslag 1 er basert på brukervennlighet og muligheter med systemet. Forslag 1 gir renovatørene informasjon om ruten, og tilbakemelding underveis dersom det blir gjort en feil. Vi har sett at det blir litt tungvint å

gå med en PDA på seg når renovatørene skal ut og inn av bilene hele tiden. Forslag 2 gjør at renovatørene enten slipper å ha noe utstyr med seg, eller kun trenger å ha med seg en IDBlue enhet. En skjerm i bilen vil dessuten være lettere å se på enn å måtte ta opp en PDA hver gang det skulle være noe.

Med tanke på hvor tungvint det er å måtte gå rundt med både PDA og IDBlue i forhold til PC i bilen og fastmontert RFID leser kan vi ikke anbefale noe annet enn å velge en løsning basert på PC i bilen og fastmontert RFID leser. Vi har også vært i kontakt med ingeniørvesenet i Kristiansand Kommune som står bak SmartSortert. Her har de gode erfaringer etter å ha innført RFID merking av avfallsdunker. SmartSortert har basert sin løsning på PC og RFID leser fastmontert i renovasjonsbilene. Informasjonen sendes automatisk over til en MSSQL database når bilen ankommer avfallsanlegget.

11 Konklusjon

I oppgaven har vi sett på hvordan RDF kan brukes for å koble sammen metadata, slik at disse kan brukes til kvalitetssikring i en case hos Agder Renovasjon. Under kartleggingen av hvilke RDF verktøy som eksisterer bestemte vi oss for å benytte Jena i denne oppgaven.

Vår erfaring med RDF er at dette er et bra rammeverk. Det er enkelt å opprette modeller, integrere modeller og hente ut informasjon. En stor fordel er muligheten for varierende antall attributter på ressursene. I en vanlig relasjonsdatabase må det opprettes en egen kolonne dersom en ny attributt skal legges til. Dette fører til unødvendig plassforbruk for de radene som ikke har informasjon om den nye attributten.

Det virker som om RDF fortsatt er på forskningsstadiet. De løsningene som finnes er ofte utviklet som en del av et forskningsprosjekt og eksisterer i tidlige versjoner. RDF er et tregere alternativ enn tradisjonelle databaseløsninger. Ulike RDF verktøy gir ulike muligheter for tilgangskontroll.

Er RDF i kombinasjon med RFID relevant for Agder Renovasjon? RDF har mange fordeler fremfor tradisjonelle systemer. Den kanskje største fordelen er de enkle integreringsmulighetene. Samtidig har RDF noen ulemper, som plassforbruk og effektivitet. Agder Renovasjon vil både med og uten RDF ha mulighet til å bedre kvalitetssikringen ved hjelp av merking av avfallsdunker. Så lenge det ikke er et stort behov for å enkelt kunne integrere flere eksterne system, vil tradisjonelle databaseløsninger være å foretrekke. Dette på grunn av eksisterende kompetanse på disse løsningene, samtidig som det er en stor jobb å legge om eksisterende løsninger til RDF. Vi mener derfor at Agder Renovasjon med dagens situasjon bør velge en tradisjonell databaseløsning.

Vi har i denne oppgaven utviklet en demonstrator som i tillegg til RDF benytter en PDA og en bærbar RFID leser. Etter å ha sett hvor tungvint det er for renovatørene å gå rundt med en PDA og IDBlue, er vi ikke i tvil om at det vil være mye bedre med fastmontert utstyr i bilene. Bærbart utstyr fører raskt til batteriproblemer. Med fastmontert PC i bilene og stasjonær RFID leser vil registreringen av dunkene bli mye enklere. Det vil også være lettere å formidle mer informasjon til renovatørene.

RFID har fungert greit i forbindelse med denne demonstratoren. Vi mener at RFID vil fungere bedre i et kvalitetssikringssystem enn strekkoder.

Referanser

[1] @semantics, http://www.asemantics.com, Sist lest: 25. mai 2005

[2] David Norheim, ”Enterprise Information Integration, Solve your ’Asemantics’ First”, http://www.asemantics.com/n/papers/Enterprise_information_integration.pdf, mai 2004

[3] Sun Kyung Kim, Byung Gon Kim, Jaeho Lee og Hae Chull Lim, ”The Strategies for Storing RDF Documents using RDBMS”, The 6th International Conference on Advanced Communication Technology, side 1027-1029, 2004

[4] Youn Hee Kim, Byung Gon Kim, Jaeho Lee og Hea Chull Lim, ”The Strategy of Transform from XML and Storage Structure base don RDB for RDF”, The 6th International Conference on Advanced Communication Technology, side 1030-1034, 2004

[5] Jeen Broekstra, Arjohn Kampman og Frank van Harmelen, ”Sesame: An Architecture for Storing and Querying RDF Data and Schema Information”, http://www.cs.vu.nl/~frankh/postscript/MIT01.pdf, 2001

[6] David Norheim, ”The Semantic Web, Summary and resources”, http://www.asemantics.com/n/papers/Semantic_Web_summary_and_resources.pdf, mai 2004

[7] Ora Lassila og Ralph R. Swick m.fl, “Resource Description Framework (RDF), Model and Syntax”, http://www.w3.org/TR/WD-rdf-syntax-971002/, 1997

[8] Tim Berners-Lee, “Weaving the Web”, Harper San Francisco, 1999

[9] W3C, “Semantic Web”, http://www.w3.org/2001/sw, Sist lest: 25. mai 2005

[10] SchemaWeb, http://www.schemaweb.info/, Sist lest: 25. mai 2005

[11] Sean B. Palmer, “A rough guide to Notation3”, http://infomesh.net/2002/notation3/, Sist lest: 26. mai 2005

[12] Jan Grant og Dave Beckett, “RDF Test Cases”, http://www.w3.org/TR/2004/REC-rdf-testcases-20040210/, Sist lest: 26. mai 2005

[13] Uche Ogbuji, “Introducing N-Triples”, http://www-128.ibm.com/developerworks/xml/library/x-think17/, 8 Apr 2003

[14] ICS-FORTH - Foundation for Research and Technology - Hellas, Institute of Computer Science, http://www.ics.forth.gr/, Sist lest: 26. mai 2005

[15] The SeRQL query language (revision 1.2), http://www.openrdf.org/doc/users/ch06.html, Sist lest: 26. mai 2005

[16] SeRQL 1.1 Query Examples, http://www.openrdf.org/sesame/serql/serql-examples.html#simple, Sist lest: 26. mai 2005

[17] HP Labs, “RDQL - RDF Data Query Language”, http://www.hpl.hp.com/semweb/rdql.htm, Sist lest: 26. mai 2005

[18] Intelli dimension, “RDFQL Database Command Reference”, http://www.intellidimension.com/?topic=/pages/rdfgateway/reference/db/, Sist lest: 26. mai 2005

[19] Eric Prud'hommeaux og Andy Seaborne, ”SPARQL Query Language for RDF”, http://www.w3.org/TR/2004/WD-rdf-sparql-query-20041012/, Sist lest: 26. mai 2005

[20] Eric Prud'hommeaux og Andy Seaborne, “SPARQL Query Language for RDF”, http://www.w3.org/TR/2005/WD-rdf-sparql-query-20050419/, Sist lest: 26. mai 2005

[21] @semantics, “SPARQL-ing days”, Monte San Savino, Tuscany, http://www.asemantics.com/n/events/apr-2005.html, 12. april 2005
[22] Drive, http://www.driverdf.org/, Sist lest: 26. mai 2005
[23] Jena, http://jena.sourceforge.net, Sist lest: 26. mai 2005
[24] Joseki, http://www.joseki.org, Sist lest: 26. mai 2005
[25] Sesame, http://www.openrdf.org, Sist lest: 26. mai 2005
[26] Aduna, http://aduna.biz/, Sist lest: 26. mai 2005
[27] On-To-Knowledge prosjektet, http://www.ontoknowledge.org/, Sist lest: 26. mai 2005
[28] NLnet, http://www.nlnet.nl/, Sist lest: 26. mai 2005
[29] Ontotext Lab, “Knowledge and Language Engineering Lab of Sirma”, http://www.ontotext.com/, Sist lest: 26. mai 2005
[30] AKT - 3store, http://www.aktors.org/technologies/3store/, Sist lest: 26. mai 2005
[31] Redland RDF Application Framework, http://librdf.org/, Sist lest: 26. mai 2005
[32] RDFStore, http://rdfstore.sourceforge.net/, Sist lest: 26. mai 2005
[33] RDFAuthor, http://rdfweb.org/people/damian/RDFAuthor/, Sist lest: 26. mai 2005
[34] IsaViz, http://www.w3.org/2001/11/IsaViz/, Sist lest: 26. mai 2005
[35] Graphviz, http://www.graphviz.org/, Sist lest: 26. mai 2005
[36] EAN Norge, http://www.ean.no/default.asp?artID=240&show=art, Sist lest: 26. mai 2005
[37] Digi.no, http://www.digi.no/php/art.php?id=109659, Sist lest: 26. mai 2005
[38] Savi Technologies, ”Part 1: Active and passive RFID: Two Distinct, But Complementary, Technologies for Real-Time Supply Chain Visibility”, http://www.autoid.org/2002_Documents/sc31_wg4/docs_501-520/520_18000-7_WhitePaper.pdf, Sist lest: 25. mai 2005
[39] Symbol, http://www.symbol.com/products/mobile_computers/mc9000-g_with_rfid.html, Sist lest: 26. mai 2005
[40] TEK Industries Inc, http://www.tekind.com/rfid/rfidpalm.htm, Sist lest: 26. mai 2005
[41] RFIDusa.com, http://rfidusa.com/superstore/product_info.php?cPath=34&products_id=233, Sist lest: 26. mai 2005
[42] Cathexis Innovations Inc, http://www.cathexis.com/products/idblue.asp, Sist lest: 26. mai 2005
[43] Socket Communications, http://www.socketcom.com/product/RF5300-542.asp, Sist lest: 26. mai 2005
[44] PDAToday, http://www.pdatoday.com/comments/1840_0_1_0_C/, Sist lest: 26. mai 2005
[45] Syscan International Inc, http://www.syscan.com/w4/it_read_mobile_e.htm, Sist lest: 26. mai 2005
[46] RFID Journal, http://www.rfidjournal.com/article/articleview/393/1/1/, Sist lest: 26. mai 2005
[47] PSI Systems AS, http://www.psi.no, Sist lest: 26. mai 2005
[48] Smart Sortert, http://www.smartsortert.no, Sist lest: 26. mai 2005
[49] Justis- og Politidepartementet, ”Intern Kvalitetssikring”, http://odin.dep.no/jd/norsk/dok/andre_dok/nou/012001-220012/hov009-bn.html, Sist lest: 26. mai 2005
[50] Politiken.dk, http://politiken.dk/VisArtikel.sasp?PageID=376578, Sist lest: 26. mai 2005

[51] ServeTheWorld, http://www.servetheworld.no, Sist lest: 26. mai 2005
[52] Ryan Lee, ” Scalability Report on Triple Store Applications”, http://simile.mit.edu/reports/stores/, Sist lest: 26. mai 2005
[53] Open Directory Project, http://dmoz.org/, Sist lest: 26. mai 2005
[54] Graham Klyne og Jeremy J. Carroll, ”Resource Description Framework (RDF): Concepts and Abstract Syntax”, http://www.w3.org/TR/rdf-concepts/, Sist lest: 26. mai 2005
[55] Shelley Powers, “Practical RDF”, O’REILLY, første utgave juli 2003
[56] Michael C. Daconta, Leo J. Obrst og Kevin T. Smith, ”The Semantic Web”, WILEY, 2003

Vedlegg

Vedlegg A – CD-Rom med kildekode